Carte d'acquisition RH_FLASH

O. Bourrion

Carte d'acquisition RH_FLASH

Table des Matières :

Table des figures :

Liste des tableaux :

1 Description

1.1 Généralités

La carte **rh_flash** a pour fonction de convertir et de stocker dans une mémoire de sortie l'image des données analogiques présentées à l'entrée et ce dans une fenêtre temporelle donnée et autour d'un signal de stop (un peu comme un oscilloscope numérique) voir Figure 1.

Elle est conçue au format VME avec connecteur auxiliaire, elle a une largeur d'une unité.

L'originalité de cette carte est de permettre une grande fréquence d'échantillonnage couplée à une fréquence de d'acquisition élevée (grâce aux codeurs flash). Pour ce faire, la carte permet de mémoriser uniquement les données jugées «utiles ». Pour ce faire elle stocke seulement les données dépassant un certain seuil, ce qui peut par exemple être utile pour voir des pics dans un spectre et les situer les uns par rapport aux autres.

Le but est de stocker (et de traiter) des données situées avant et après l'arrivée d'un signal de stop (d'où l'importance de la profondeur mémoire).

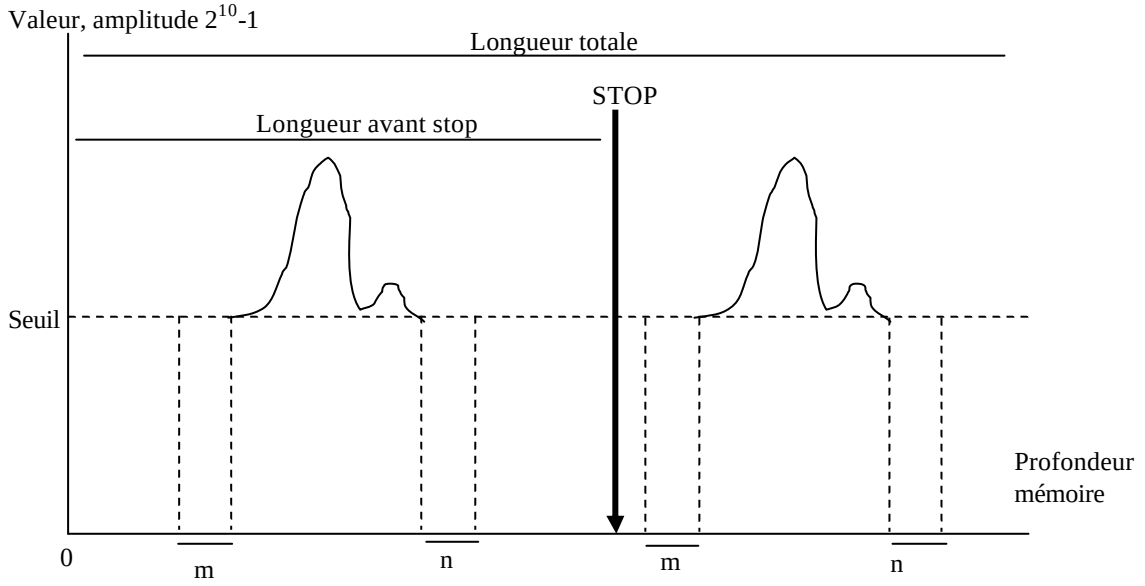


Figure 1 : fenêtre d'acquisition temporelle

On définit un seuil, et tout pic dépassant ce seuil sera réellement stocké en mémoire et finalement accessible par le bus VME. Les valeurs du pic lui-même seront stockées ainsi que les m valeurs précédentes et les n valeurs suivantes.
Évidemment, si le seuil est 0, cela revient à désactiver le système de traitement et toutes les données seront stockées.

1.2 Fonctionnement d'un canal

1.2.1 Explication fonctionnelle

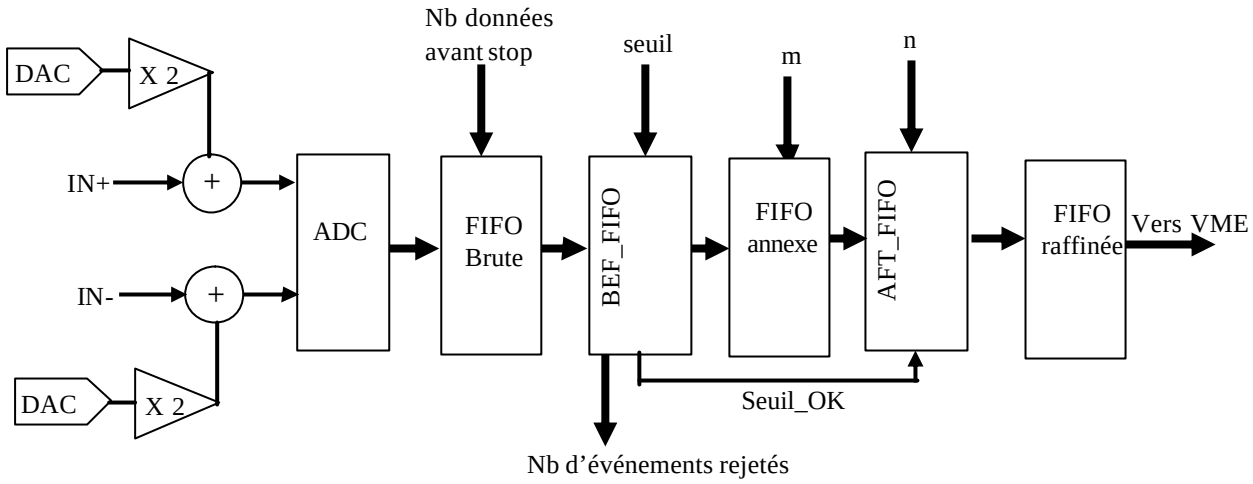


Figure 2 : Vue schématique d'un canal

Pour permettre de fonctionner avec différents formats de signaux analogiques en entrée, ceux-ci sont conditionnés à l'aide de 2 DAC permettant d'effectuer un décalage et d'un ampli op fonctionnant en additionneur. Après cette étape, les signaux sont encodés grâce à l'ADC sur chaque front montant de l'horloge (chaque bloc ci-dessus est synchrone).

La première FIFO sert à stocker les données brutes en provenance du flash ADC de façon à garder en mémoire les données avant le trigger car elle fonctionne en buffer circulaire. C'est cette FIFO qui est effectivement programmée en série avec le nombre de données avant stop.

Le module de traitement BEF_FIFO sert à relire les données brutes et à marquer toutes les données dépassant le seuil programmé avant de les stocker dans la mémoire FIFO annexe (c'est effeet une FIFO intermédiaire dans le traitement). C'est ce qui permet de stocker les données avant pic, car cette FIFO annexe fonctionne elle aussi en buffer circulaire et est programmée avec la valeur m. Ainsi, lorsqu'on entre dans une zone pic, les données sont retardées de m cycles, alors que l'information seuil_OK est directement transmise au module de traitement AFT_FIFO. Le module est ainsi informé de l'entrée prochaine dans un pic et active ainsi le stockage en FIFO_raffinée. Dans le module BEF_FIFO, il y a un compteur d'événements rejetés, qui est incrémenté à chaque fois que le signal stop alors qu'un traitement est déjà en cours.

Le module de traitement AFT_FIFO a aussi d'autres rôles, ainsi lorsque l'arrivée imminente d'un pic est détectée, ce module insère une marque relative de pic, c'est-à-dire qu'il indique le nombre de cycle écoulés depuis le début de la fenêtre. Aussi à la fin de la fenêtre, il ajoute une marque de fin de fenêtre à la suite des données.

Au final, les données stockées dans la FIFO raffinées ne sont que celles supérieures au seuil plus m données avant et n données après. Dans le cas d'une fenêtre sans pic, il n'y aura qu'une marque de fin de fenêtre dans le buffer de sortie.

Il est à noter que l'horloge utilisée pour cadencer l'acquisition est générée sur la carte.

1.2.2 Façade avant

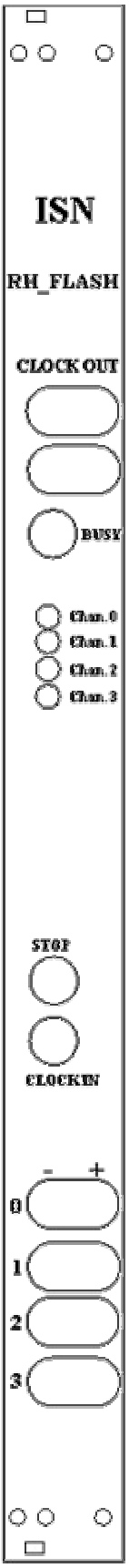


Figure 3 : vue de la façade

1.2.3 Position des composants

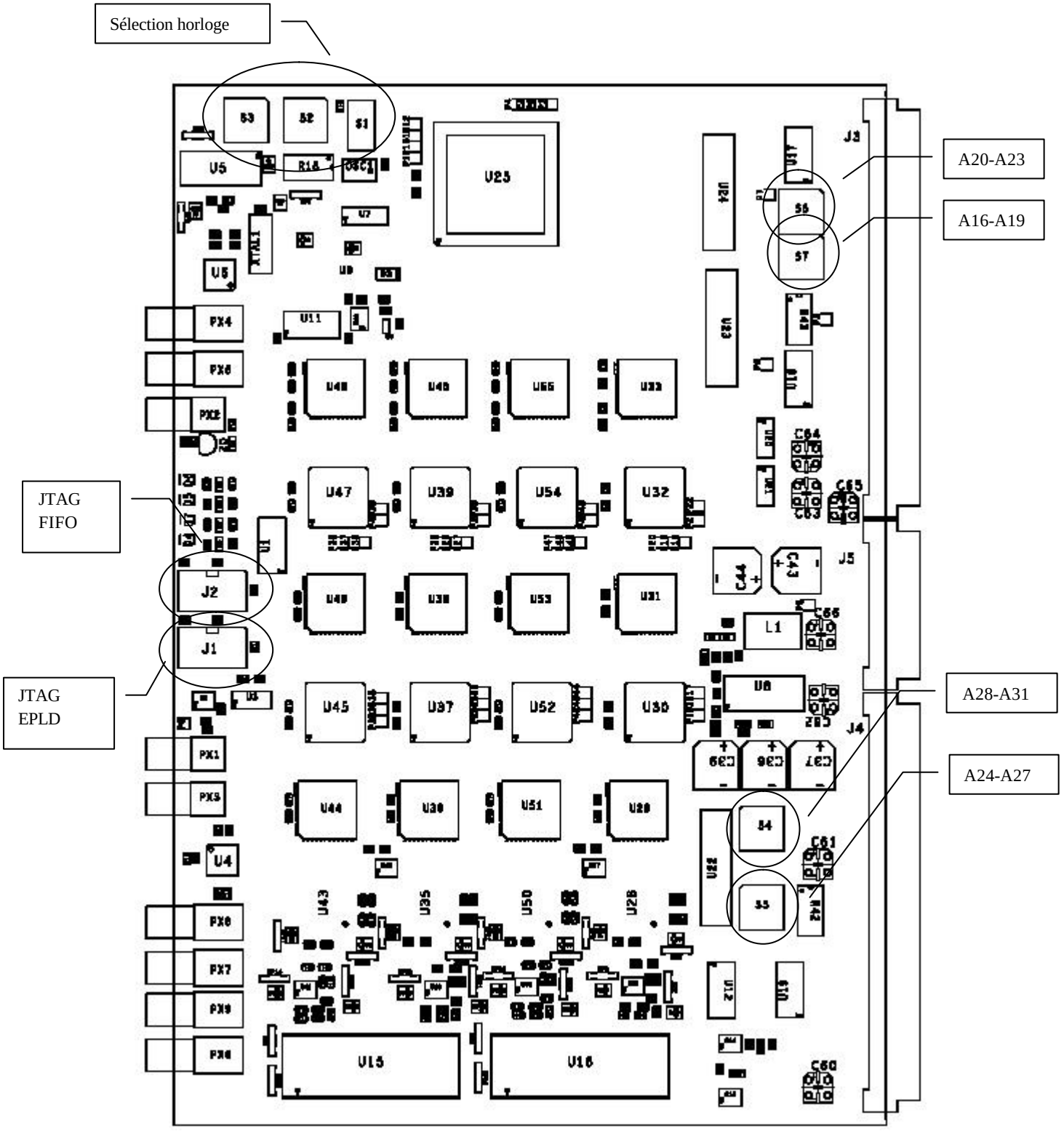


Figure 4 : Position des roues codeuses et connecteurs

2 Spécifications

2.1 Générales

- Carte VME au format CERN ;
- Résolution 10 bits ;
- 4 voies indépendantes, individuellement sélectionnables ;
- Fréquence de fonctionnement programmable entre 30MHz et 100MHz par pas de 1MHz ;
- Taille de chaque buffer de sortie : 32768 mots (permet 320 µs d'acquisition à 100MHz) ;
- Possibilité de stocker plusieurs fenêtres dans le buffer de sortie ;
- Taille de la fenêtre : 5 <taille< 32767 ;
- Données d'encadrement : 5< m, n<255 ;
- SNR (***S**ignal to **N**oise **R**atio*) en unipolaire entrée continue : 59,6 dB → ENOB (***E**ffective **N**umber **O**f **B**it*)=9,6
- SNR avec sinus à 1MHz 1Vpp en unipolaire : à remplir
- Non linéarité intégrale : +1,3 LSB max de 12bits (données constructeur du DAC) ;
- Non linéarité différentielle : +1 LSB max de 12bits (données constructeur du DAC) ;

2.2 Entrées

Analogiques :

- 0V .+2V, single ended positif ;
- 0V .-2V, single ended négatif ;
- -1V .+1V, single ended impulsion bipolaire ;
- -1V.0V sur entrée IN- et 0V.+1V sur IN+ pour différentiel unipolaire ;
- -0,5V .+0,5V, différentiel impulsion bipolaire ;
- impédance d'entrée de 50? ;

STOP : Entrée NIM, largeur min 6 temps de cycle ;

CLOCK_in : Entrée LVTTL, avec 30MHz<F<100 MHz ;

Toutes ces entrées ont impédance d'entrée de 50? .

2.3 Sorties

CLOCK_out(3 ..0) : sorties LVTTL avec un jitter RMS inférieur à 50ps. Les différentes sorties sont des copies du même signal. Le skew entre ligne est inférieur à 200 ps.

BUSY : sortie NIM, asynchrone par rapport aux signaux d'horloge CLOCK_in. =1 si traitement en cours ou si l'un buffers de sortie contient 31kmots (max=32k).

2.4 Consommation

+5V : 4A ;
-5V : inférieur à 100 mA ;

3 Explication du design de chaque canal

Se référer en annexe pour voir le schéma.

3.1 Conditionnement analogique

À l'entrée, les signaux analogiques passent par un ampli-op monté en soustracteur (car le DAC fournit des tensions négatives) de gain 1 sur l'entrée analogique et gain 2 sur l'offset. Celui est un amplificateur *rail to rail* ayant une bande passant de 300 MHz. Chaque entrée est terminée par une résistance de 50? au plus près des amplificateurs. Il est à noter que les alimentations des ampli-op sont très fortement filtrées.

Attention, il n'a pas été placé de filtre anti-repliement en entrée de l'ADC, de cette façon toute les impulsions peuvent être vues, même si elles risquent d'être déformées.

3.2 ADC

L'AD9433 est un codeur flash 12 bits ayant pour fréquence de fonctionnement max 105MHz. Il encode la différence des signaux présentés à ses entrées AIN à chaque basculement de ENCODE. Le fonctionnement de l'ADC est légèrement détourné, normalement il faut présenter un signal différentiel dont l'amplitude doit être 2* 1V p-p autour de la tension de mode commun qui est +4V. Mais dans le cas d'un fonctionnement unipolaire avec le câblage réalisé on applique 2 V p-p sur une seule entrée. L'ADC fonctionne dans ces conditions, mais au détriment d'une plage utile légèrement réduite, environ 1,9 V p-p au lieu de 2 V p-p.

Le signal appliqué sur ENCODE est un signal PECL différentiel. Le buffer a été choisi pour que son temps de traversée (600 ps max) ne cause pas de desynchronisation entre les divers composants numériques du canal. Aussi lors du routage, la ligne d'horloge alimentant chaque ce buffer a été placée de façon a être plus courte que les autres de 400 ps environ.

Budget temps : un SY100ELT22 (tpd max =600 ps). Le tco max de l'AD9433 est de 5,5 ns, le skew max entre ligne de 250ps, le tds min de la FIFO est de 2 ns donc :

0,6+5,5+0,25+2=8,35 ns, laisse une marge de 1,65 ns.

Rappel de la formule du SNR (d'après MAXIM) du au jitter :

$$SNR = \frac{1}{s_{RMS} * 2\Pi f}$$

Où T_{RMS} est le jitter ;
F est la fréquence du signal analogique.

Or le bruit de quantification est SNR=1.76 + 6.02N dB, soit :

- Pour 10 bits → 61,96 dB
- Pour 12 bits → 74 dB

3.3 Voyant d'activation

Pour ne pas charger la sortie de l'EPLD de traitement, un montage a été réalisé avec un MOS pour avoir un courant suffisamment important dans la LED verte.

3.4 FIFO BRUTE

Elle est câblée en buffer circulaire, grâce au retour du signal PAF (Programmable Almost Full) sur le signal REN (Read Enable). Ainsi lorsque la FIFO est presque pleine, le signal de lecture est actif, ce qui fait que autant de mots écrits sont lus ➔ le niveau reste constant à la valeur programmée.

La programmation de PAF est faite de façon sérielle, de manière à éviter de compliquer le chemin de données et de garder des performances raisonnables.

Cette FIFO est autorisée à se remplir lorsque WEN est à l'état bas, cela arrive 2 ou 3 cycles d'horloge après l'activation du canal (RUN=1).

Cette FIFO a une taille de 32768 mots.

3.5 Module de traitement BEF_FIFO

Sa fonction principale est de marquer les données appartenant à la fenêtre et celle dépassant le seuil programmé.

3.5.1 Interface

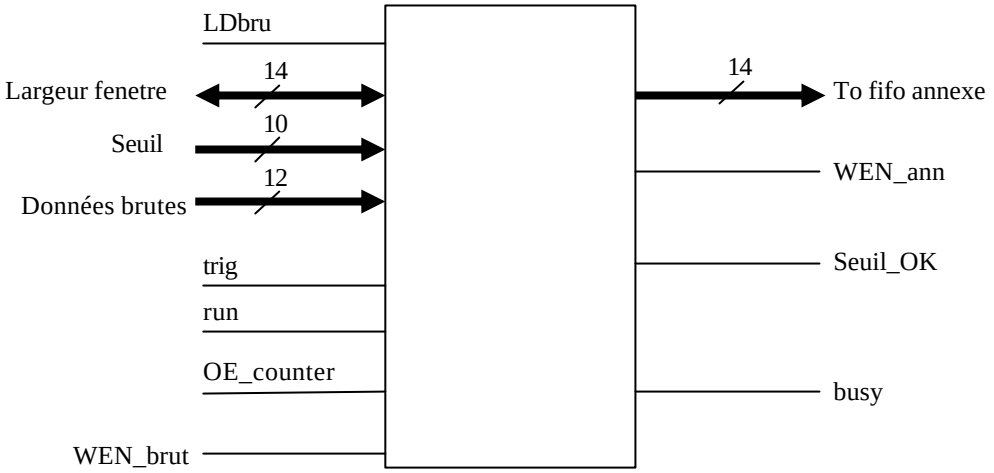


Figure 5 : interface de BEF_FIFO

Budget : 59 IO

- **LD_bru** : autorise le chargement des données de configuration **Largeur_fenetre** et **seuil** ;
- **Largeur_fenetre** : signal contenant la demi largeur de fenêtre ;
- **Seuil** : contient le seuil de référence appliqué au comparateur sur 10 bits ;
- **Donnees_brutes** : ce sont les données encodées sur 12 bits en provenance de la FIFO_BRUTE ;
- **Trig** : c'est le signal de stop resynchronisé sur l'horloge de la carte ;
- **Run** : c'est le signal d'activation de la voie ;
- **OE_counter** : permet d'autoriser la sortie du compteur d'événement rejeté sur le bus largeur_fenêtre ;
- **WEN_brut** : c'est le signal qui valide l'écriture dans la FIFO_BRUTE ;
- **To_FIFO_annexe** : contient les données encodées par l'ADC, plus 2 bits indiquant si la donnée est dans la fenêtre et si la donnée est supérieur au seuil ;
- **WEN_ann** : c'est le signal qui valide l'écriture dans la FIFO_ANNEXE ;
- **Seuil_OK** : signal indiquant si la donnée envoyée vers FIFO_ANNEXE est supérieure au seuil ;
- **Busy** : signal actif lorsque un traitement est en cours, c'est-à-dire après un stop et pendant le balayage de la fenêtre ;

3.5.2 Synoptique du module de traitement

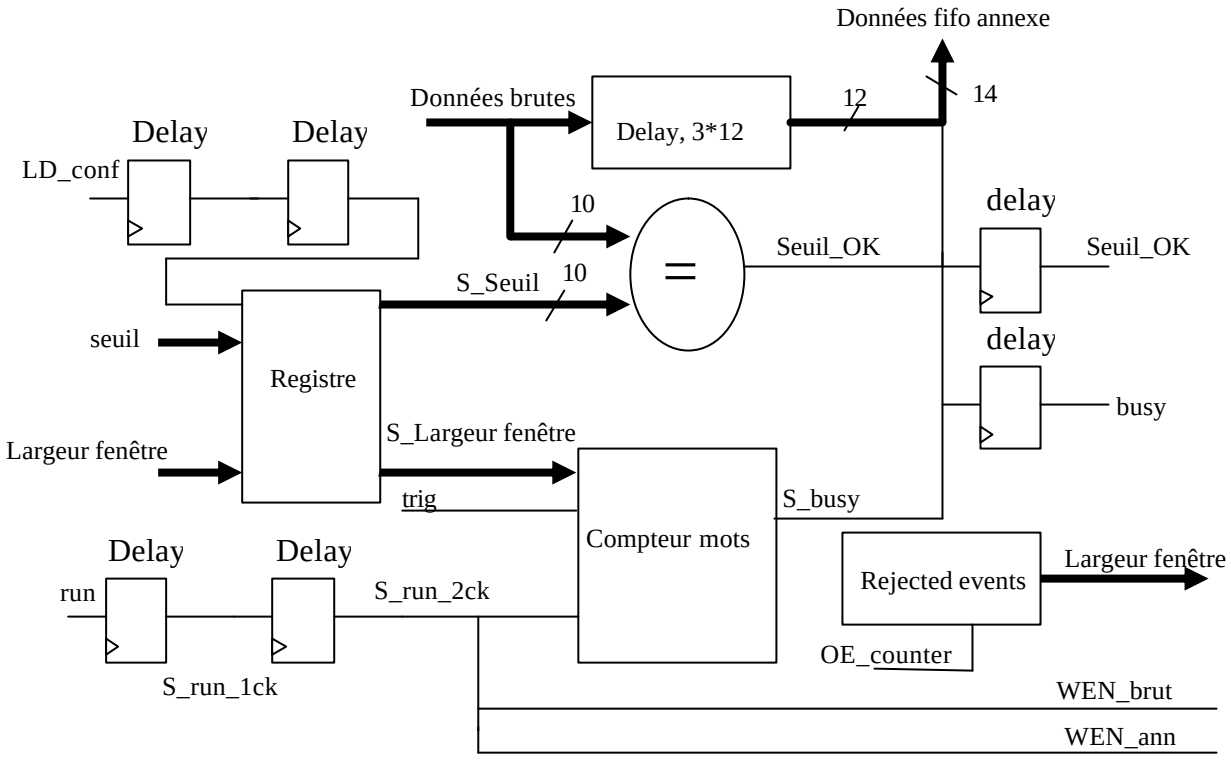


Figure 6 : synoptique de BEF_FIFO

WEN_brut et WEN_ann sont la copie de run après 2 bascules de resynchronisation préchargée à 1. RUN doit être resynchronisé car ce signal provient de l'EPLD interf_VME qui lui fonctionne de manière asynchrone au reste de la carte.

Le bus de données brutes est de taille 12 bits. L'extension à 12 bits en sortie n'a pas été faite car l'ADC est donné avec un ENOB (*Effective Number Of Bit*) compris entre 10,9 et 11,1 en typique sur la gamme de fréquence analogique visée. La comparaison se fera toujours sur les 10 MSB.

Le compteur de mots indique si l'on est dans la fenêtre à traiter (les mots avant et après stop à traiter). Le bit **BUSY** signalant que l'on est dans la fenêtre est ajouté aux octets de données.

La comparaison qui se déroule en ligne ainsi que l'écriture dans la FIFO annexe. Toutes les données ayant une valeur supérieure au seuil sont marquées avec un MSB à 1 avant d'être écrites dans la mémoire.

Les lignes à retard (registres) pour les données servent à permettre la comparaison en ligne (retard de 3 coups d'horloge).

Bitmap des mots en FIFO annexe :

13	12	11	…	0
Fenêtre_OK	Seuil_OK	Donnée encodée sur 12 bits		

Tableau 1 : Format des données en sortie de BEF_FIFO

- **Fenêtre_OK** : 1 quand la donnée fait partie de fenêtre de travail ;
- **Seuil_OK** : =1 quand la donnée marquée est supérieure au seuil.

Le signal trig (stop) est resynchronisé sur la carte par 2 bascules D en série, puis distribué vers toutes les voies. Ceci pour éviter la métastabilité et pour être sûr d'avoir le même fonctionnement sur les 4 voies.

3.5.3 Comparateur 10 bit pipeliné

La comparaison fonctionne toujours et ne peut être désactivée.
La comparaison 10 bits à été découpée en comparaisons 4 bits pour pouvoir pipeliner le design. Ainsi, la comparaison commence par les MSB. Les égalités sont testées pour vérifier que l'on a bien un strictement supérieur (et pas un supérieur ou égal), car dans le cas contraire, il faut aller comparer les bits de poids plus faible (cascade) pour trancher.

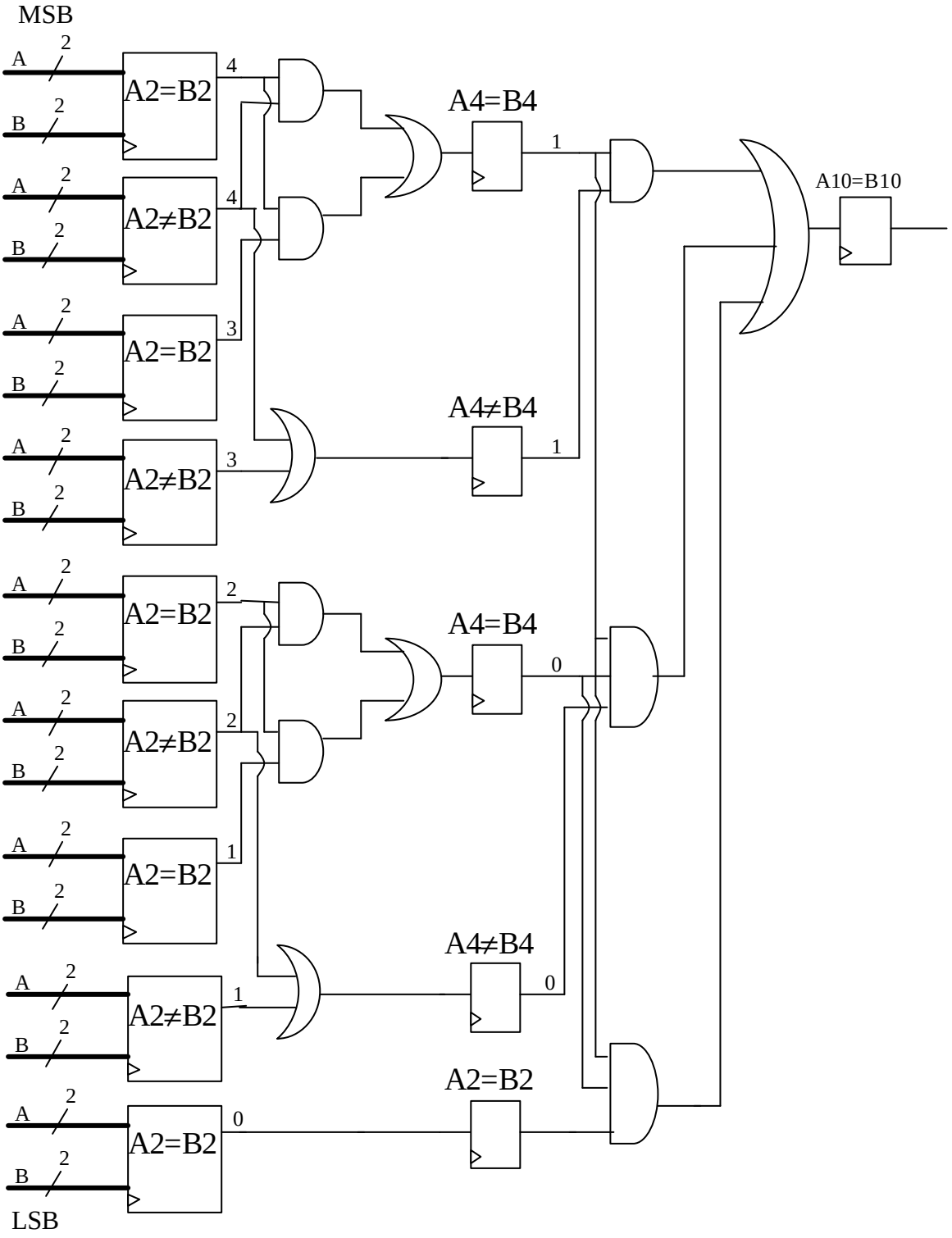


Figure 7 : schéma du comparateur 10 bits

Budget : 15 bascules.

3.5.4 Compteur de mots

Permet de gérer une fenêtre de 32k mots. Cela implique un compteur et un comparateur pipeliné sur 15 bits.

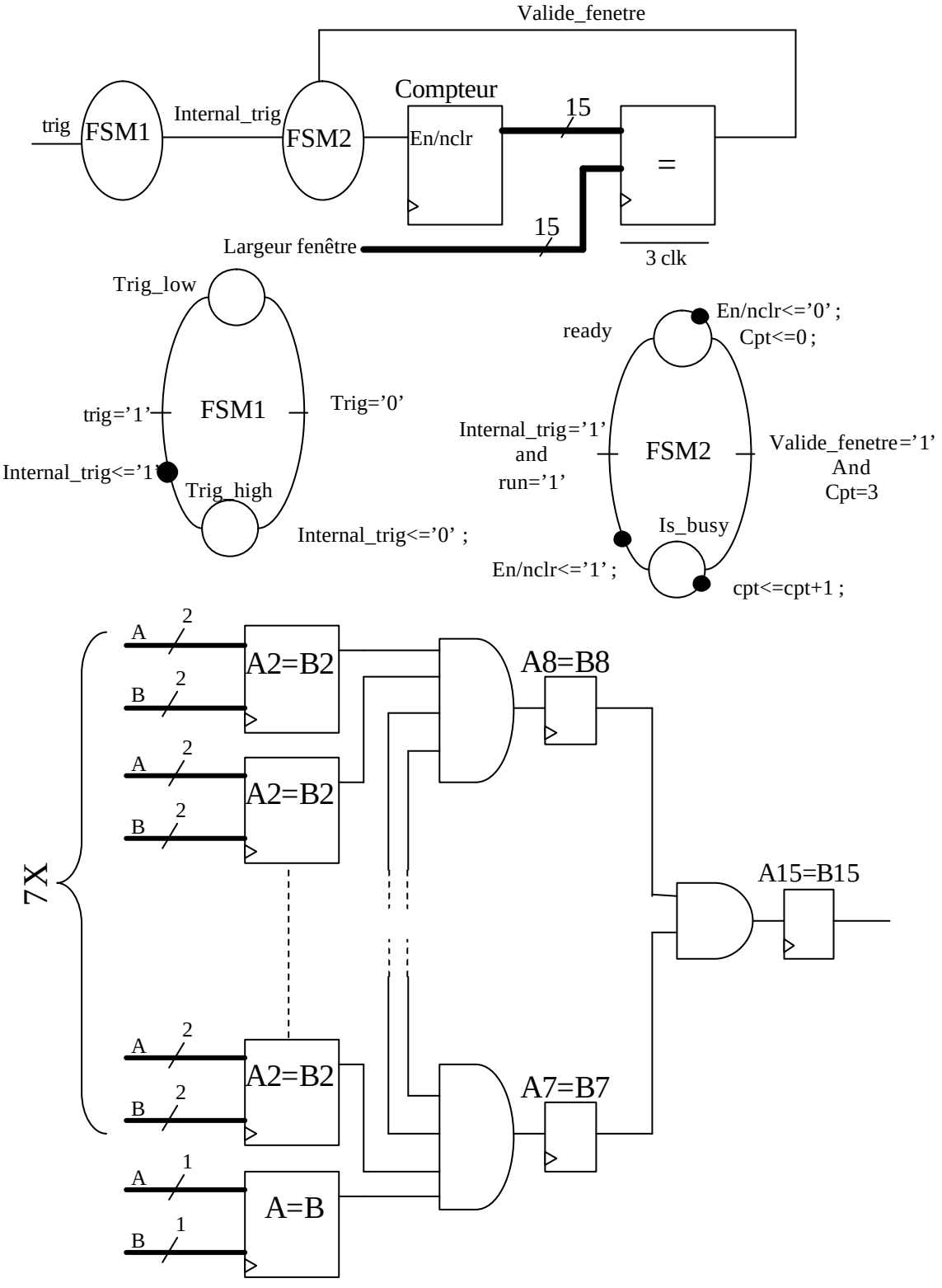


Figure 8 : Compteur de mots

Budget :

- comparateur : 11 bascules ;

- compteur : 15 bascules ;
- FSM1 + FSM2 : 4 bascules ;
- Cpt3 : 3 bascules + 2 comp =5

Total 35 bascules.

La machine d'état **FSM1** sert à faire une « détection de front » et ainsi évite un redéclenchement intempestif sur une impulsion de stop trop longue.

La machine d'état **FSM2** sert à autoriser le marquage des données appartenant à la fenêtre si un **stop** valide a été détecté et si la voie est activée (**run**=1). Elle sert aussi à prendre en compte la latence du module de comparaison d'égalité. En effet, à cause de la latence du comparateur, la fenêtre doit avoir une largeur d'au moins 3 mots.

3.5.5 Compteur d'événements rejetés

Il faut ajouter un compteur d'événements rejetés qui sera incrémenté lorsqu'un trigger arrive alors que le canal est occupé (BUSY).

Total EPLD :

- Delay : 39 bascules ;
- Comparateur seuil : 15 bascules ;
- Registre + anti meta : (10+15+2)=27 bascules ;
- Compteur mot / valide fenetre : 35 bascules ;
- Compteur événement rejeté : 15 bascules ;

Total =131 bascules

3.6 FIFO ANNEXE (ou intermédiaire)

Fonctionnement quasiment identique à FIFO BRUTE, page 10. La seule différence porte sur la taille maxi de la FIFO, celle-ci a une taille de 2048.

3.7 Module de traitement AFT_FIFO

Sa fonction est de situer temporellement les pics par rapport au début de la fenêtre et donc par rapport aux autres voies si elles sont programmées de manière identique. Il sert aussi à valider l'écriture des données à conserver dans la FIFO raffinée.

3.7.1 Interface

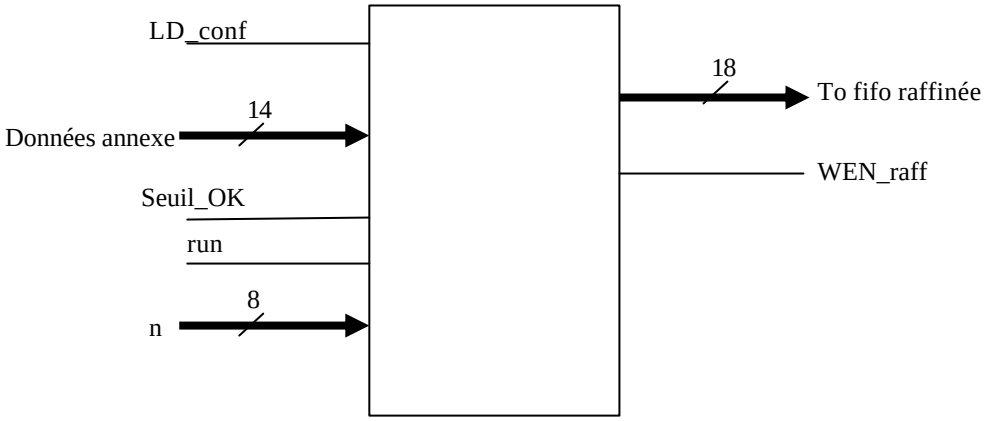


Figure 9 : Interface du module AFT_FIFO

Budget : 44 IO

- **LD_conf** : valide le chargement du bus n ;
- **Donnees_annexe** : bus contenant les données codées sur 12 bits en provenance de la FIFO ANNEXE. Voir Tableau 1 : Format des données en sortie de BEF_FIFO, page 12 ;
- **Seuil_OK** : entrée indiquant si des données supérieures au seuil sont en transit dans la FIFO annexe ;
- **Run** : signal activant la voie ;
- **N** : nombre de données à conserver après un pic ;
- **To_fifo_raffinee** : données à écrire en FIFO raffinée ;
- **WEN_RAFF** : ordre d'écriture dans la FIFO ;

Format des données présentées sur le bus **To_fifo_raffinee** :

17	16	15	…	0
Fin_de_fenêtre	Pos_relative / donnée	Donnée		

Tableau 2 : Format des données en sortie de AFT_FIFO

- **Fin_de_fenêtre** : =1 si le mot indique une fin de fenêtre, dans ce cas la donnée portée ne signifie rien ;
- **Pos_relative / donnée** : =1 quand donnée contient une marque de position relative codée sur 16 bits. Dans le cas contraire donnée porte une donnée ADC codée sur les 12 bits de poids faibles ;

3.7.2 Synoptique

Tous les signaux entrant (excepté n) passent par des registres non représentés.

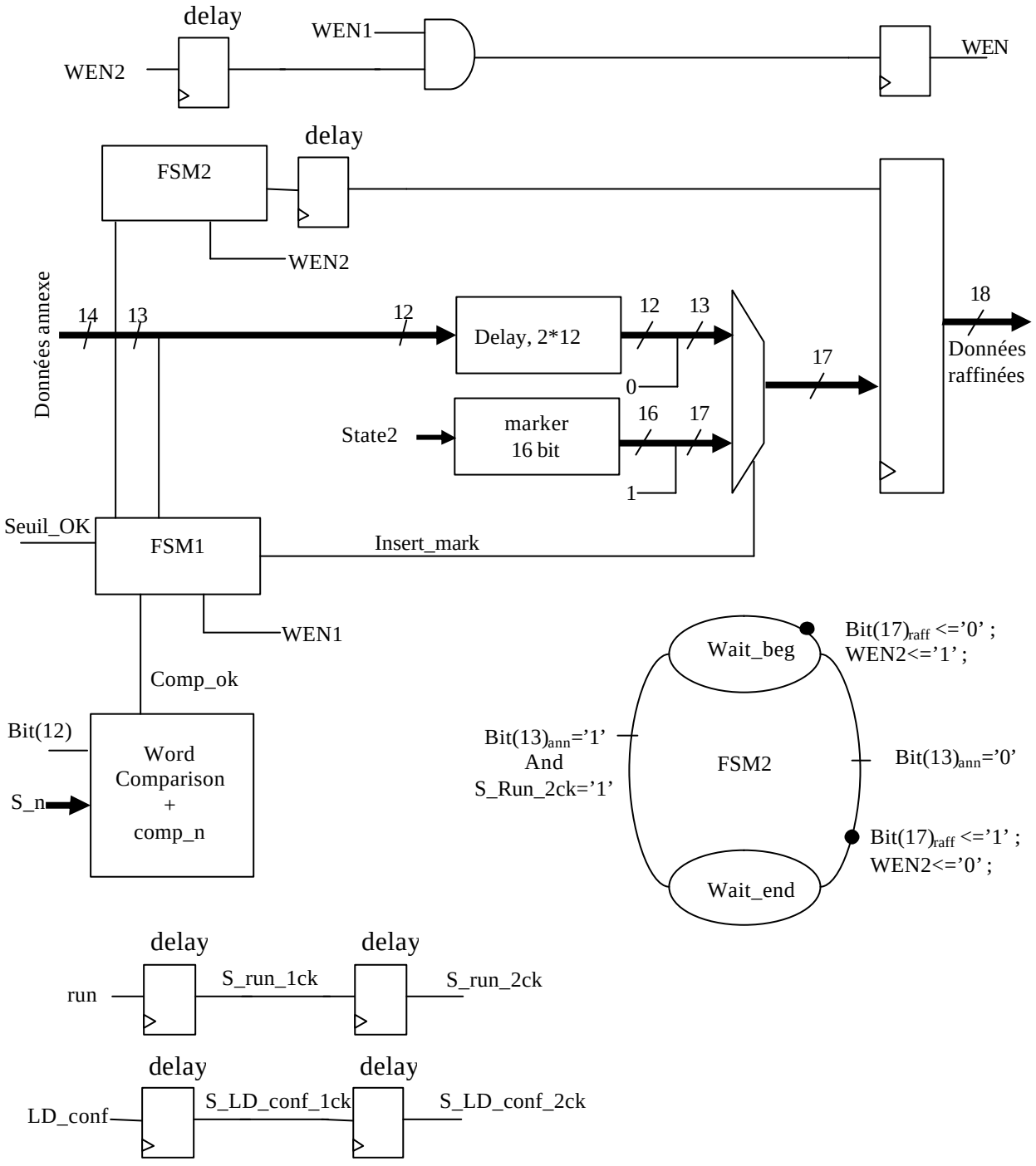


Figure 10 : Synoptyque du module de traitement AFT_FIFO

La **FSM2** sert à insérer un MSB à 1 dans la FIFO raffinée (sert à marquer la fin de la fenêtre). En fait la **FSM2** détecte la sortie de fenêtre en testant le bit 13 du bus **donnee_annexe** (voir Tableau 1 : Format des données en sortie de BEF_FIFO, page 12)

La ligne à retard sert à permettre l'insertion des marqueurs avant que la donnée ADC soit vraiment présente.

Word_comparison sert à compter le nombre de données après le pic ; il fonctionne sur le même principe que le compteur de mots (voir Figure 8 : Compteur de mots, page 15) et il est re-déclenchable. Du coup n doit avoir une valeur minimum de 3 ou 4. Il est remis à zéro dans le pic car il doit compter les données après le pic.

La **FSM1** sert à insérer un repère relatif devant chaque pic, de façon à pouvoir les situer les uns par rapport aux autres. En dehors de la fenêtre de travail, elle est désactivée. La position relative insérée est la position par rapport au début de la fenêtre et non pas par rapport au stop. La donnée utilisée est fournie par le compteur dénommé **marker**.

Marker est un compteur 16 bits remis à 0 par la **FSM2**, c'est-à-dire en dehors de chaque fenêtre de travail.

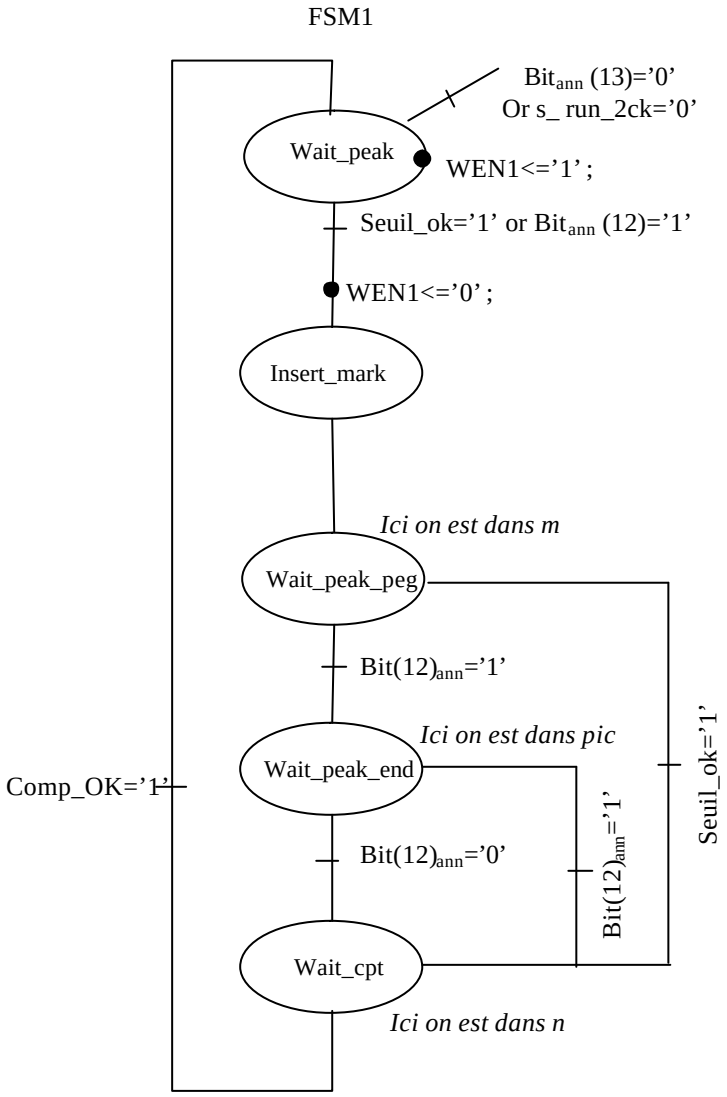
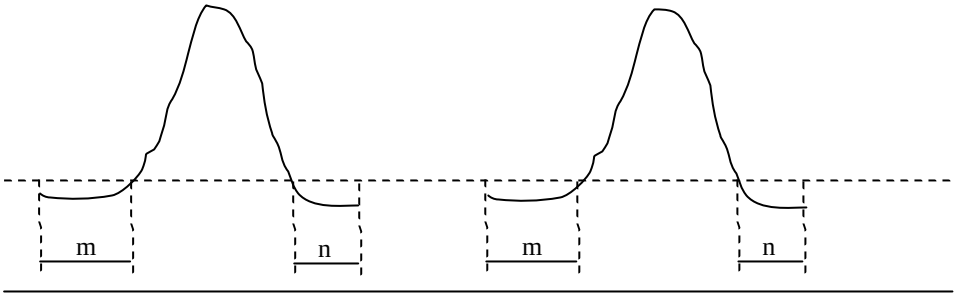
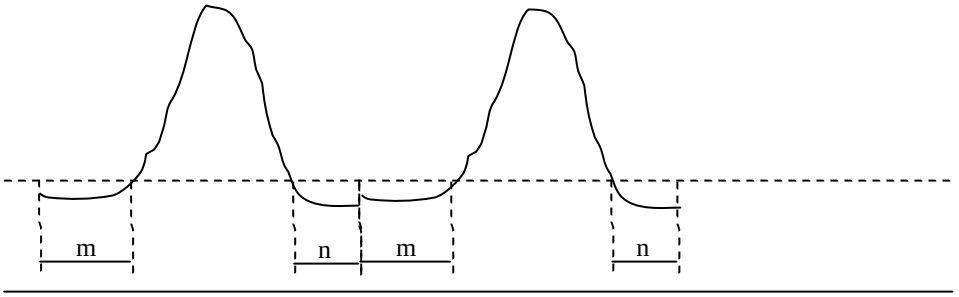


Figure 11 : Module de marquage de pic

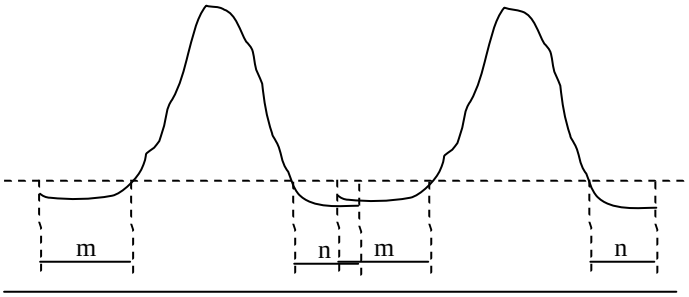
2 Pics distincts :



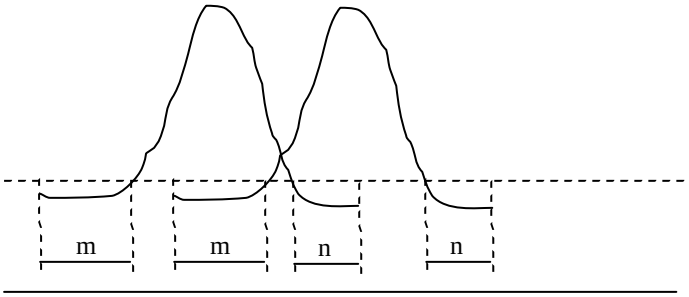
2 pics à la limite du re-déclenchement :



2 pics faisant re-déclencher la machine FSM1 :



Ou :



Il faut n>m. Dans le cas contraire si les pics sont trop proches (m<écartement<n-m), le deuxième est vue trop tard.

Budget :

- FSM1 : 4 bascules ;

- FSM2 : 2 bascules ;
- Delay : 24 bascules ;
- Input reg : 12 bascules ;
- Output reg : 19 bascules ;
- Comp_n : 7 + 5 = 12 bascules ;
- Marker : 16 bascules ;
- Divers : 5 bascules ;

Total : 95 bascules.

3.8 FIFO_RAFFINEE

Cette FIFO permet de stocker les données utiles et d'interfaces deux mondes d'horloge différents, le coté acquisition (cadencé jusqu'à 100 MHz) et le coté VME (cadencé à 32 MHz). Cette FIFO n'est pas prévue pour être programmée, le prépositionnement par défaut qui est PAE=1023 et PAF= 31745 sera utilisé. Le bus de sortie des FIFO RAFFINEE des 4 voies est commun, la sélection se fait en passant le bus à lire en basse impédance.

4 Explication du design commun

4.1 Chaînes JTAG

Deux chaînes JTAG sont implantées sur la carte :

- Une permettant la programmation des EPLD ;
- Une permettant le test des FIFO ;

Chaque bus passe par un buffer permettant d'adapter les tensions de 2,5V au standard LVTTL. Pour la chaîne EPLD, l'EPLD d'interface VME est placé en tête car il fonctionne en 3V3, ainsi c'est lui qui pilotera les autres EPLD fonctionnant en 2,5V (la compatibilité ne fonctionne pas nécessairement dans les 2 sens).

Numéro d'ordre dans la chaîne	identifiant
1	Interf_VME
2	AFT_FIFO voie 0
3	AFT_FIFO voie 1
4	AFT_FIFO voie 2
5	AFT_FIFO voie 3
6	BEF_FIFO voie 3
7	BEF_FIFO voie 2
8	BEF_FIFO voie 1
9	BEF_FIFO voie 0

Tableau 3 : Chaîne EPLD

Numéro d'ordre dans la chaîne	identifiant
1	FIFO_RAFFINEE, voie 0
2	FIFO_RAFFINEE, voie 1
3	FIFO_RAFFINEE, voie 2
4	FIFO_RAFFINEE, voie 3
5	FIFO_ANNEXE, voie 3
6	FIFO_ANNEXE, voie 2
7	FIFO_ANNEXE, voie 1
8	FIFO_ANNEXE, voie 0
9	FIFO_BRUTE, voie 0
10	FIFO_BRUTE, voie 1
11	FIFO_BRUTE, voie 2
12	FIFO_BRUTE, voie 3

Tableau 4 : Chaîne FIFO

4.2 Mise en forme du trigger

Le signal stop est un signal NIM, il faut donc le convertir en TTL. Pour ce faire, un comparateur rapide (AD8561) compare le signal d'entrée à une référence de -0,45V. La sortie de ce signal est envoyée vers un jeu de 2 bascules qui resynchronise ce signal avant de le distribuer au 4 voies. Cela évite tout risque d'interprétation différente entre les voies (pour cause de métastabilité).

4.3 Mise en forme du signal BUSY

Le signal est converti de LVTTL vers NIM grâce à un montage constitué de 2 transistors, BFT92 et BC547C. C'est un montage en collecteur ouvert.

4.4 Horloges

Chaque fin de ligne est adaptée par un réseau de thévenin de 100? , et les lignes sont tracées de façon à avoir des délais identiques (excepté les 4 lignes alimentant les buffers PECL, voir 3.2, page 9)

4.4.1 « 100 MHz »

Comme la gigue de l'horloge doit être la plus faible possible (*jitter*), l'horloge est générée sur la carte à l'aide d'un synthétiseur d'horloge programmable très performant. Les coefficients du synthétiseur sont chargés à la sortie du reset VME.

L'horloge générée par ce composant est envoyé vers un composant spécialisé dans la distribution d'horloge, c'est-à-dire qu'il a un fan out de sortie très élevé et que les temps de propagation entre les diverses sorties sont très proches (skew faible). Quatre sorties sont générées permettant de cadencer 4 cartes RH_FLASH de façon rigoureusement synchrone.

Pour obtenir la fréquence désirée en sortie, utiliser les formules suivantes :
Dans la gamme 25→ 50 MHz : Fout=M/8 ;
Dans la gamme 50→ 100 MHz : Fout=M/4 ;
Attention, il faut diviser M par 2 avant de l'appliquer sur les roues codeuses car M0 est câblé à zéro, et les 8 bits des roues sont connectés en fait aux bits (8..1).

Le tableau suivant peut être utilisé pour éviter les erreurs :

Fout	N	M	S1	S3	S2
30	16	240	1	7	8
32	16	256	1	8	0
34	16	272	1	8	8
36	16	288	1	9	0
38	16	304	1	9	8
40	16	320	1	A	0
42	16	336	1	A	8
44	16	352	1	B	0
46	16	368	1	B	8
48	16	384	1	C	0
50	8	200	0	6	4
52	8	208	0	6	8
54	8	216	0	6	C
56	8	224	0	7	0
58	8	232	0	7	4
60	8	240	0	7	8
62	8	248	0	7	C
64	8	256	0	8	0
66	8	264	0	8	4
68	8	272	0	8	8
70	8	280	0	8	C
72	8	288	0	9	0
74	8	296	0	9	4
76	8	304	0	9	8
78	8	312	0	9	C
80	8	320	0	A	0
82	8	328	0	A	4
84	8	336	0	A	8
86	8	344	0	A	C
88	8	352	0	B	0
90	8	360	0	B	4
92	8	368	0	B	8
94	8	376	0	B	C
96	8	384	0	C	0
98	8	392	0	C	4
100	8	400	0	C	8

Tableau 5 : Sélection de fréquence

4.4.2 « 32 MHz »

Il y aussi une horloge de 32 MHz qui est générée sur la carte pour l'interfaçage VME et la lecture des FIFO. Elle passe aussi par un buffer d'horloge avant distribution.

4.4.3 Alimentations

4.4.3.1 Consommations max estimées

4.4.3.1.1 Par voie

composant	I(5V)	I(3V3)	I(2V5)

EPLD bef_FIFO	-	-	284
EPLD aft_FIFO	-	-	243
FIFO_brute	-	-	90
FIFO_annexe	-	-	90
FIFO_raffinee	-	-	90
Régulateur 2V5	10	-	-
Flash_ADC	285	14	-
analogique	30	-	-
SY100ELT22	30	-	-
Adaptation PECL	24	-	-
Led	-	15	-
Total / Alim	**379**	**29**	**797**
Total toute alim	**1205**		

Les formules ci-dessous viennent de la note d'application n°74 d'**Altera.** Elles permettent d'estimer la consommation des EPLD.
ICCINT = (A × MCTON) + [B × (MCDEV – MCTON)] + (C × MCUSED × **fMAX** × **togLC**)
The parameters in this equation are:
MCTON = Number of macrocells with the Turbo BitTM option turned on, as reported in the MAX+PLUS II Report File (**.rpt**)
MCDEV = Number of macrocells in the device
MCUSED = Total number of macrocells in the design, as reported in the Report File
fMAX = Highest clock frequency to the device
togLC = Average percentage of logic cells toggling at each clock (typically 12.5%)

EPLD bef_FIFO : 220mA (d'après calcul 129LC, tog à 12,5%) + 64 mA = 284mA
EPLD aft_FIFO : 205mA (d'après calcul 110LC, tog à 12,5%) + 38 mA=243mA

Conso FIFO = 0,7*100 (ICC1) + 100*0,01*2,5*8=90 mA (formules IDT)

4.4.3.1.2 En commun

composant	I(5V)	I(3V3)	I(2V5)
Clock buffer(tree)	-	400	-
Epld VME	-	356	500
PLL	200	25	-
Clock_buffer (out)	-	215	-
Clock 30M	-	165	-
3*74AC521	-	516µA=0,5m	-
Jtag translateur	-	15,5m	-
74LVC74A	-	20m	-
AD8561	6	-	-
Pull up pll	-	3	-
MPC905 (max)	-	45	-
oscillateur	-	15	-
Regulateur (2V5)	-	10	-
Regulateur (3V3)	-	10	-
Max6301	-	4µ	-
PS74lvch162245 (5)	-	11	-
IDT74FCT164245(5)	-	50	-

Dac7625	4	-	-
Pullup decod add	-	6	-
74F07 (2)	90 (max)	-	-
Total / alim	**300**	**1327**	**500**
total	**2147**		

Conso clk buff MPC941 : I = ICCq + VCC*F*(27*Cpd) + P*Ioh (=Iol) =
5 +3,3*100M*27*10p + 25*24mA=5+90+300=400mA (si ligne 100?)

74AC521 (unité): icc+Cpd*F*vcc=40 µA + 40p*1M*3V3=40µ+132µ=172µA
Cpd en picofarad, F en mégahertz (soit une commuation par µs)
Jtag(unité) : 1 pull up + 1 transistor en conduction = 0,3+3m=3,3m
Conso 74lvc74 : I=vcc*Cpd*fi +Cl*Vcc*f0= 3,3*30*100+8*0,001 (1 trig par ms)=10000µA = 10 mA

74F07 : 32mA (actif)
74LVCH16245 = 20µ + 16*40p*1M*3.3=2132 µ=2,2mA
capa cpd + capa charge, 1 fois toute les µs

L'estimation donne (pour un DC/DC à 80 %) :
I=(4*797+500)/(2*0,8)+1327+300+4*(29+379)=5564 mA max

4.4.3.1.3 Choix de Design

Pour l'alimentation 2V5, un convertisseur DC/DC a été implanté (EL7556). Une précaution toute particulière a eu lieu pour le choix des capas de forte valeur, elles sont données avec un très faible ESR (Equivalent Serie Resistor) pour une fréquence relativement importante.

Pour l'alimentation 3V3, un régulateur linéaire a été implanté. Une diode y est implantée tête bêche pour éviter la destruction du régulateur lors de la coupure d'alimentation.

4.5 Fonction de RESET

Le reset principal est assuré par le reset du VME, cependant un superviseur de reset a été implanté pour surveiller le 2V5, ainsi que pour permettre un contrôle du reset par l'EPLD interf_VME. La sortie du superviseur passe un buffer de distribution.

4.6 Fonction d'offset (DAC)

Les DAC implantés sont des quadruples DAC 12 bits avec chargement parallèle. Ils fournissent une tension comprise entre 0V et -2V5. Pour se faire, il a fallu adjoindre une référence de tension négative. Cela a été réalisé avec un ampli-op monté en intégrateur et une référence de tension en se basant sur la datasheet de l'ADR291 de chez Analog Device.
Les DAC sont câblés en écriture seule et en mode transparent, c'est-à-dire dès que CS remonte le bus de données est transféré directement vers la conversion.

4.7 Interface VME

Pour interfacer avec le VME, des buffers translateur de niveau ont été utilisés car l'EPLD n'est pas tolérant au 5V. Aussi dans un souci d'économie des entrées-sorties, des comparateurs 8 bits ont été utilisés pour tester les adresses et le mode d'accès VME.

4.7.1 Interface

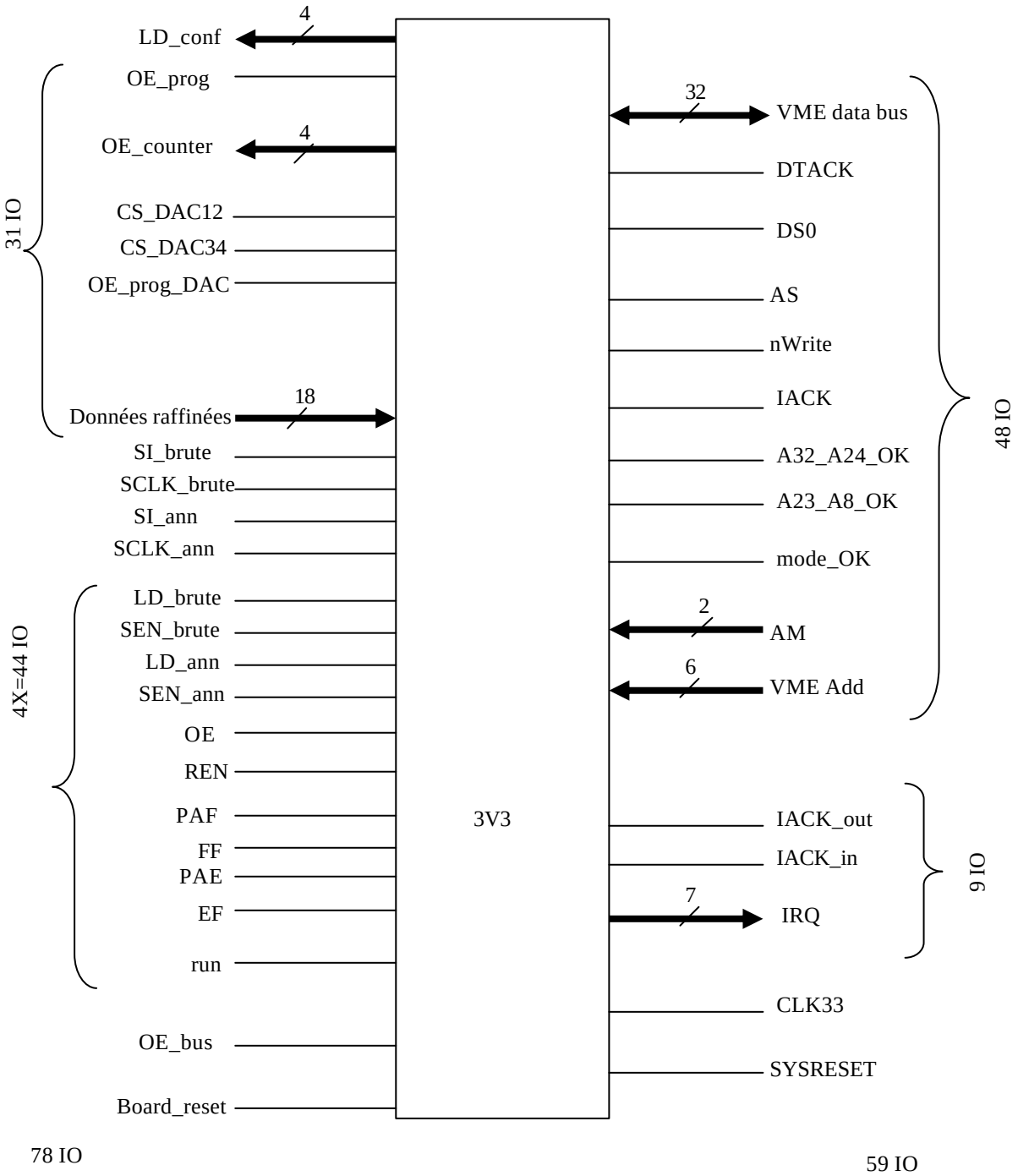


Figure 12 : interface de interf_VME

Mode_OK vérifie que AM[3..0] = $(1001)_{bin}$ et que LWORD='0' et que A[7]=$(0)_{bin}$ et DS1=$(0)_{bin}$

- **LD_conf** : autorise le chargement des paramètres (largeur_fenêtre, seuil et n) présenté sur le bus VME dans les modules de traitement ;
- **OE_prog** : utilisé en conjonction avec **LD_conf**, valide le buffer transmettant le bus de données VME vers les modules de traitement ;
- **OE_counter** : permet le passage en basse impédance de la sortie événements rejetés du module **BEF_FIFO** sélectionné ;
- **CS_DAC12** : Permet la sélection et l'écriture des 4 DAC des voies 1 et 2 ;
- **CS_DAC34** : Permet la sélection et l'écriture des 4 DAC des voies 3 et 4 ;
- **OE_PROG_DAC** : utilisé en conjonction avec **CS_DAC12** ou **CS_DAC34**, valide le buffer transmettant le poids fort du bus de données VME vers les DAC ;
- **Donnees_raffinees** : C'est le bus de données partagé entre les 4 voies ;
- **SI_brute** : sortie série de données permettant la programmation des FIFO brutes ;
- **SCLK_brute :** sortie horloge de cadencement des données séries pour les FIFO brutes ;
- **SI_ann** : sortie série de données permettant la programmation des FIFO annexes ;
- **SCLK_ann :** sortie horloge de cadencement des données séries pour les FIFO annexes ;
- **LD_brute** : Signal validant le chargement des données série dans la FIFO brute sélectionnée ;
- **SEN_brute** : Signal validant le décalage des données séries dans la FIFO brute sélectionnée ;
- **LD_ann** : Signal validant le chargement des données série dans la FIFO annexe sélectionnée ;
- **SEN_ann**: Signal validant le décalage des données séries dans la FIFO annexe sélectionnée ;
- **OE, REN** : Permettent la lecture de la FIFO raffinée sélectionnée ;
- **PAF, PAE, EF, FF** : signaux d'état de chaque FIFO raffinée ;
- **RUN** : signaux d'activation individuels de chaque voie ;
- **OE_BUS** : validation du bus VME principal ;
- **Board_reset** : signal permettant de « reseter » toutes les voies d'acquisition ;
- **A32_A24_OK** : signale que la partie du bus d'adresse 32..24 est reconnue ;
- **A23_A8_OK OK** : signale que la partie du bus d'adresse 23..8 est reconnue ;

4.7.2 Synoptique

La création d'un chip select interne appelé CS_int simplifie la lecture. Son équation est :

CS_int=(A32_A24_OK xnor AM[5..0]= « 00 ») and A24_A8_OK and Mode_OK and IACK and DS= « 00 »

Aussi CS_int passe par 2 bascules pour être resynchronisé, du coup cela laisse 2*32 ns pour le décodage et pour la stabilisation des signaux.

4.7.2.1 Module numéro de carte

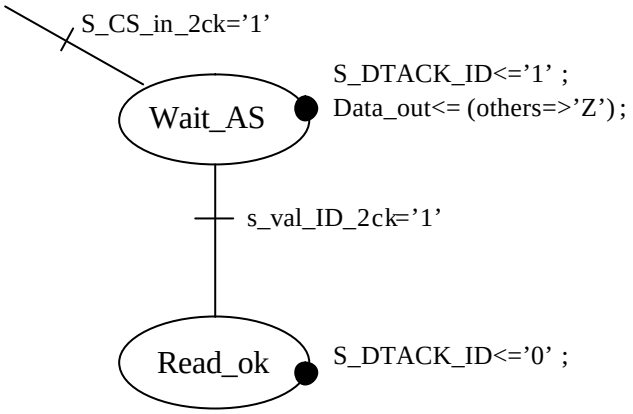


Figure 13 : FSM IDCODE

```
s_val_ID<='1' when (VME_ADD(6 downto 1)="000000" and nwrite='1') else '0';
```

2 bascules

4.7.2.2 Module mise en route acquisition

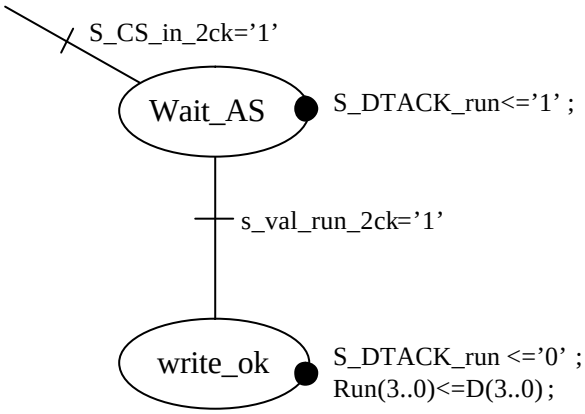


Figure 14 : FSM validation

```
s_val_run<='1' when (VME_ADD(6 downto 1)="000010" and nwrite='0') else '0';
```

2 bascules

4.7.2.3 Module de configuration des FIFO

Attention, lors de la mise en configuration des FIFO, la carte ne doit pas être en mode run.

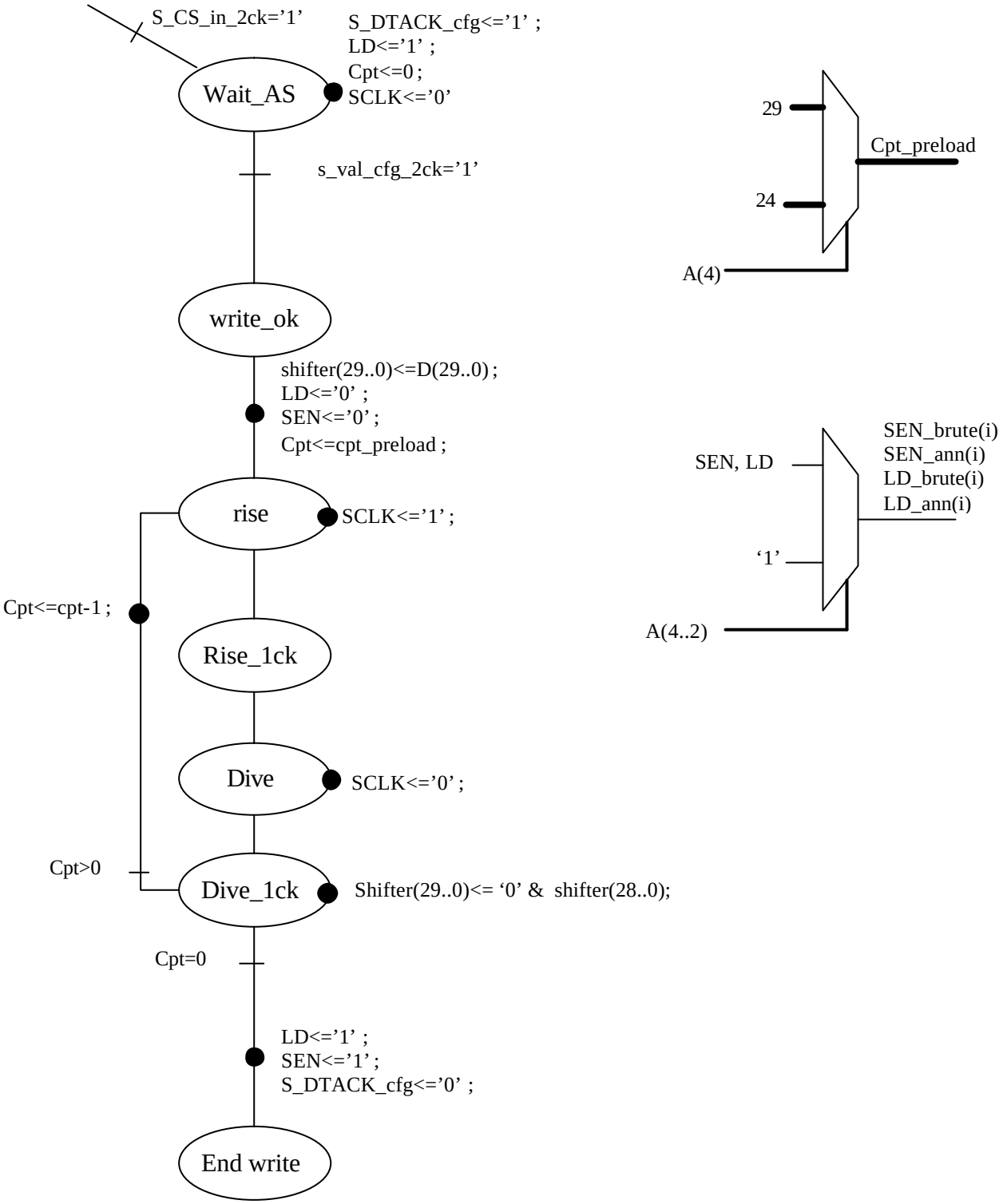


Figure 15 : FSM de configuration des FIFO

```
s_val_cfg<='1' when (VME_ADD(6 downto 5)="01" and nwrite='0' and
VME_ADD(1)='0') else '0';
```

45 bascules
La même machine d'état pilote les 8 lignes comme suit, LD et SEN sont aiguillés vers la FIFO appropriée grâce aux bits d'adresse nécessaire. SCLK et SI sont copiés vers toutes les sorties. L'état bas et l'état haut de SCLK doivent durer plus de 45 ns.
Il est à noter qu'il y a un seul registre à décalage pour les 8 FIFO, dans un souci d'économie de ressources.

4.7.2.4 Module de configuration DAC

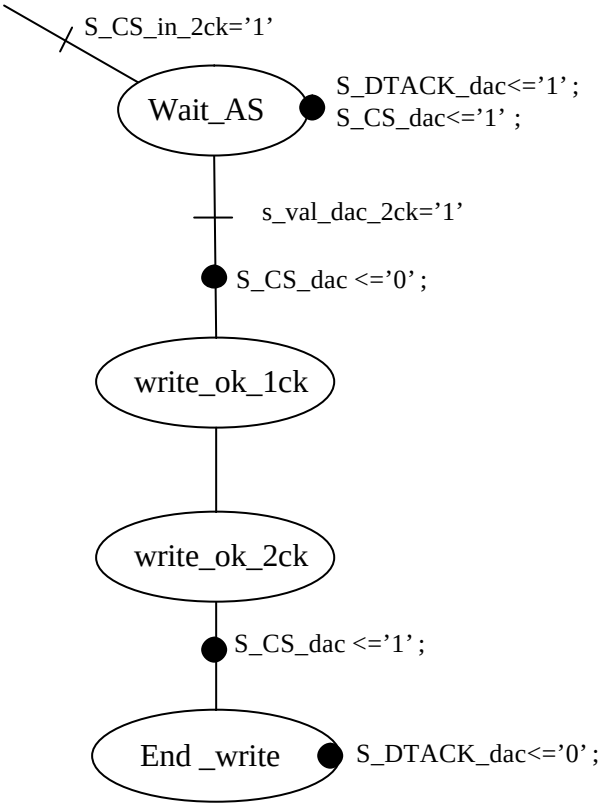


Figure 16 : FSM de programmation DAC

```
s_val_dac<='1' when (VME_ADD(6 downto 3)="0001" and nwrite='0' and
VME_ADD(1)='0') else '0';
```

Ensuite le signal s_CS_dac est aiguillé en fonction de addcfg(0) (qui est un version resynchronisé de VMEADD(2)).
Comme CS_dac dure au moins 2 périodes, le temps minimum spécifié pour les DAC est respecté.

4.7.2.5 Module de configuration fenêtre, seuil et n

Après que LD_conf soit repassé à 0 il faut attendre 3 clk avant de signaler le DTACK, car une fréquence de 25 MHz sur les données, il faut 3ck pour mémoriser les données (anti métastabilité) et une impulsion sur LDconf supérieure à 40 ns.

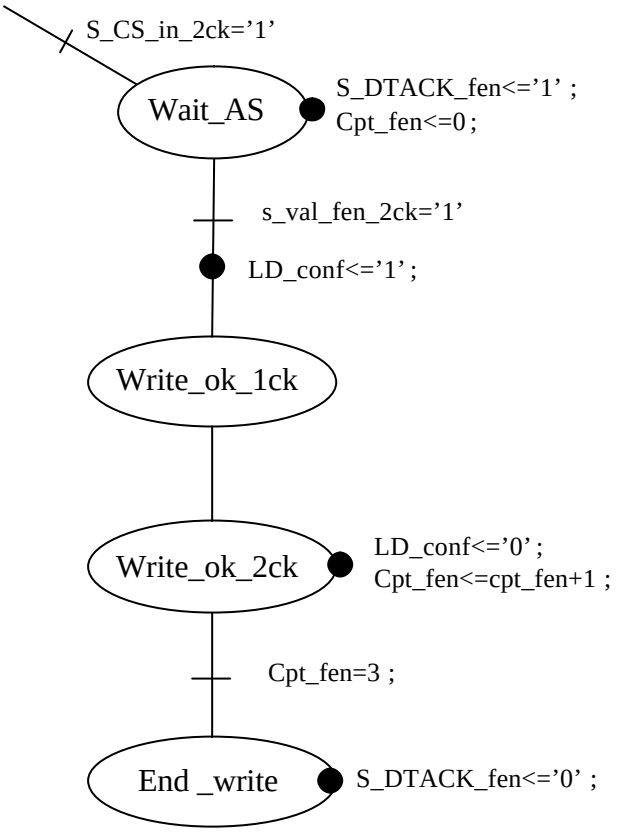


Figure 17 : FSM de configuration fenêtre, seuil, n

```
s_val_fen<='1' when (VME_ADD(6 downto 4)="001" and nwrite='0' and
VME_ADD(1)='0') else '0';
```

LD_conf sera aiguillé grâce aux bits A(2..1).
5 bascules

4.7.2.6 Module de registre de sortie

Si le MSB de la donnée lue dans la FIFO raffinée est un 1, il s'agit d'une fin de fenêtre sinon si l'avant dernier bit est à 0, il s'agit d'une donnée, sinon il s'agit d'une marque relative :

- 00 : donnée sur 10 bits ;
- 01 : marque relative (16 bits) ;
- 1X: fin de fenêtre ;

Une lecture de fenêtre commencée sur une voie doit être impérativement terminée avant de basculer sur une autre voie.

Lorsqu'une data est trouvée, il n'est pas nécessaire de vérifier l'EF (Empty Fifo) jusqu'à tomber de nouveau sur une fin de fenêtre.
Dès l'entrée dans un pic, il faut extraire les données par trois. Si lors de la lecture, un mot n'étant pas une donnée est lu, il faut mémoriser le fait qu'un pic est en cours. Ainsi lors de la lecture VME suivante, on envoie la marque de fin de pic (pas de lecture FIFO). Et à la lecture suivante on envoie le mot trouvé (début de pic ou fin de fenêtre). C'est pour ne pas se perdre dans cette machine qu'il ne faut changer de voie sans avoir extrait la fenêtre complète.

EF est aiguillé par un multiplexeur synchrone utilisant une version de VME_ADD resynchronisée. De la même manière REN et OE_fifo sont aiguillés.

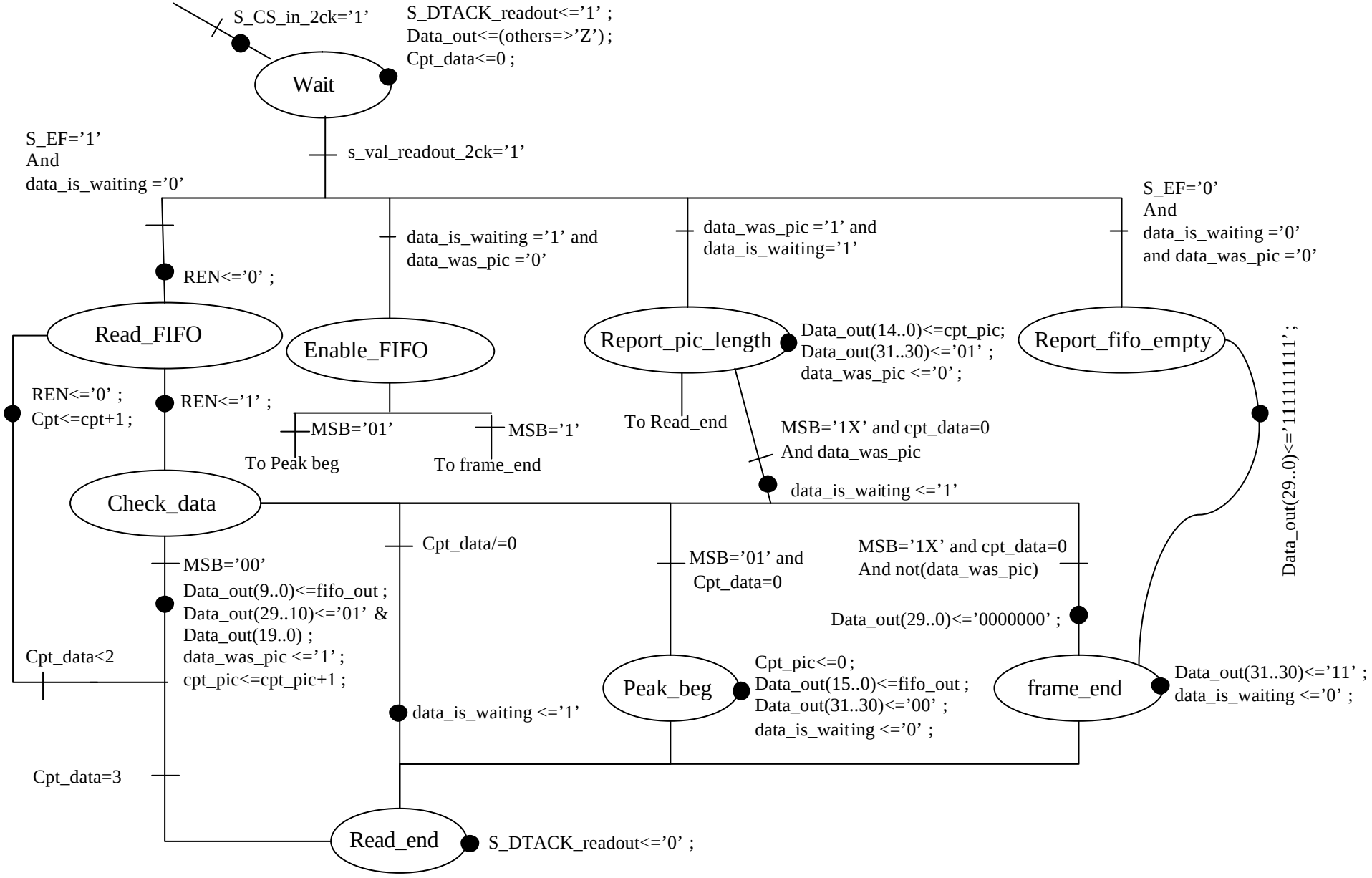


Figure 18 : FSM registre de sortie

```
s_val_readout<='1' when (VME_ADD(6 downto 4)="100" and nwrite='1' and
VME_ADD(1)='0') else '0';
```

Pour que ce module fonctionne correctement, le système doit tourner à au moins 30 MHz !!

4.7.2.7 Module compteur événements rejetés

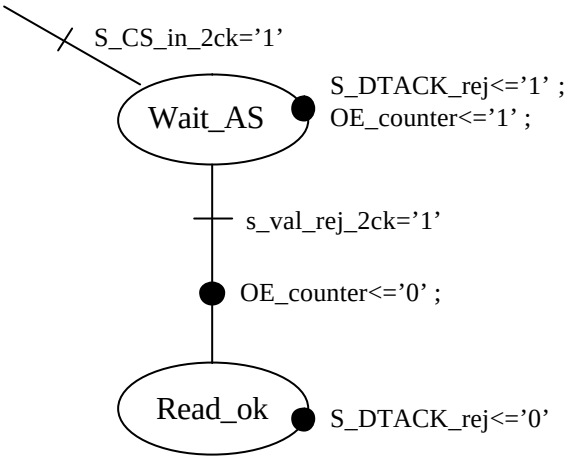


Figure 19 : FSM de lecture du nombre d'évenements rejetés

```
s_val_rej<='1' when (VME_ADD(6 downto 4)="101" and nwrite='1') else '0';
```

4.7.2.8 Module de gestion d'interruptions

4.7.2.8.1 Validation

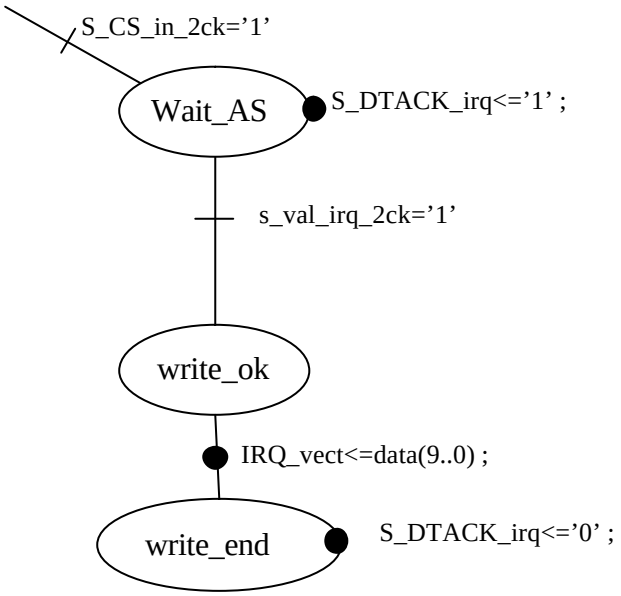


Figure 20 : FSM de validation d'interruption

```
s_val_irq<='1' when (VME_ADD(6 downto 1)="110000" and nwrite='0') else '0';
```

4.7.2.8.2 Chaînage des IT

Un retour est volontairement ajouté sur IACKIN pour avoir le temps de faire le traitement avant de propager vers IACKOUT

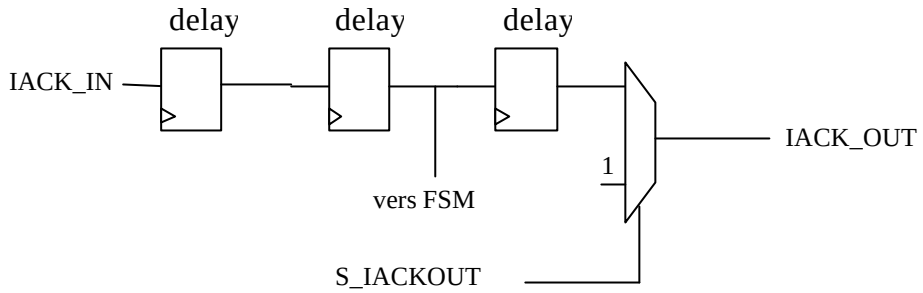


Figure 21 : gestion du chaînage d'IT

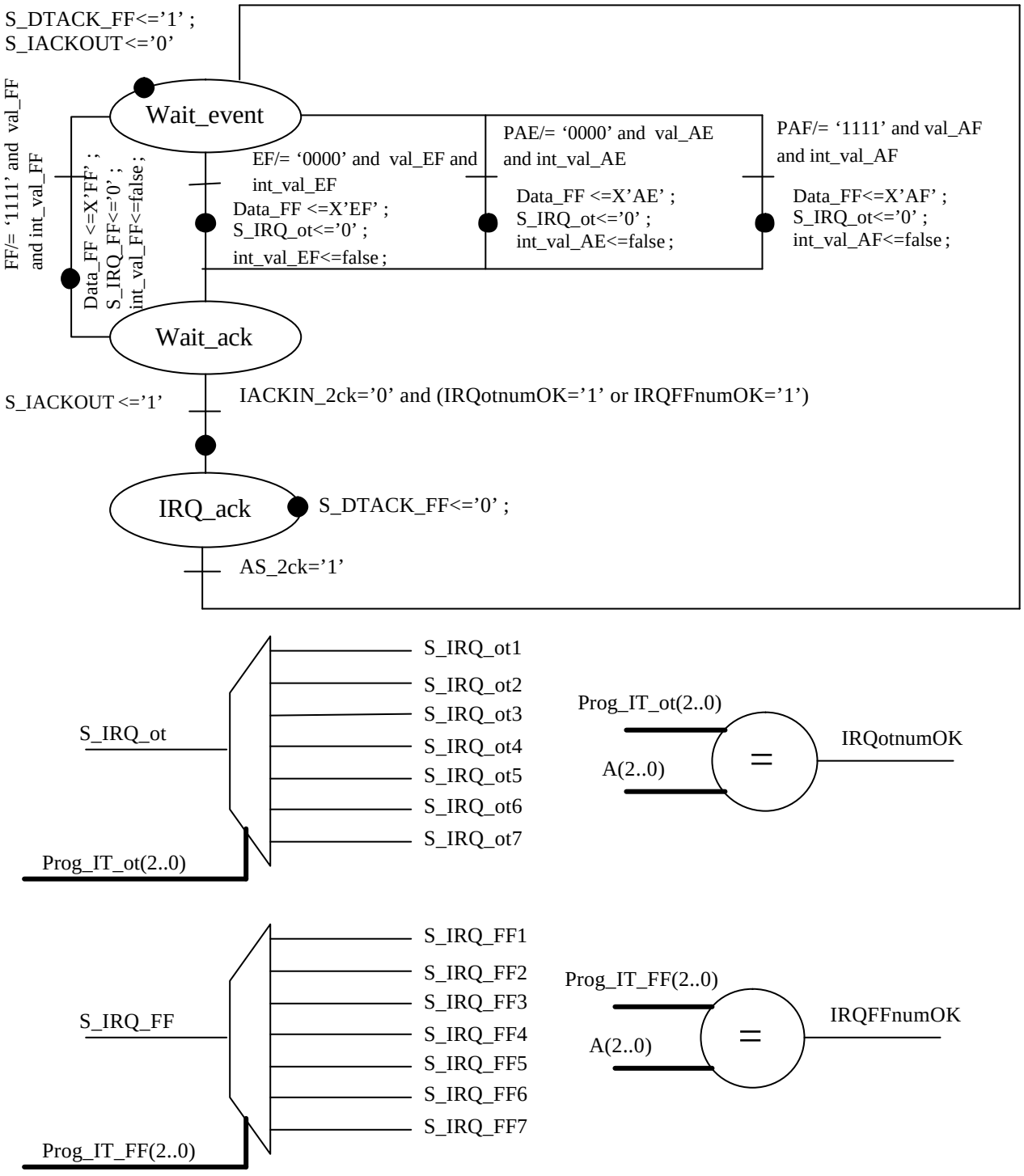


Figure 22 : gestion du cycle d'interruption

Lorsque la machine de gestion des interruptions détecte une interruption, elle invalide automatiquement l'interruption (en affectant int_val_XX à false), de cette manière le rebouclage en IT est évité. Donc après avoir traité toutes les IT d'une carte, il faut revalider toutes les IT. Lors de l'écriture dans le registre de validation, toutes les int_val_XX sont affectées à true.

Toutes les sorties IRQ et DTACK sont des sorties en collecteur ouvert grâce aux composant 74F07.

4.7.2.9 Module de reset soft

Ce module génère un niveau celui-ci en retour contrôle un MOS qui va court-circuiter la résistance du pont dans le montage superviseur de reset. En effet pour que celui-ci voit le court-circuit, l'impulsion sur le superviseur doit durer au moins 1 ms et l'impulsion sur le monostable doit durer au moins 125 ns.

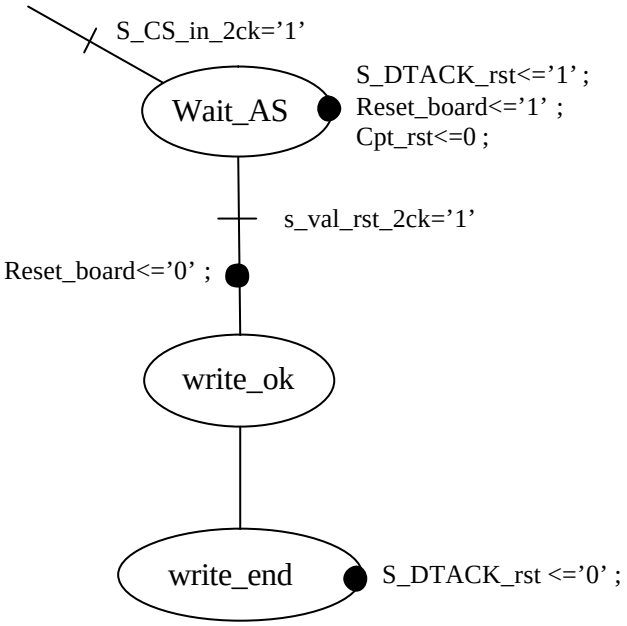


Figure 23 : FSM de reset

```
s_val_rst<='1' when (VME_ADD(6 downto 1)="111000" and nwrite='0') else '0';
```

5 Adressage

Celles-ci sont données de A31 à A1. A0 est supposée constamment nul. (Donc il faut multiplier par 2 les adresses suivantes.

5.1 Modificateur d'adresse reconnus

Les AM (Address Modifier) reconnus sont :

- 39 : accès aux données en mode utilisateur standard (A24) ;
- 09 : accès aux données en mode utilisateur étendu (A32) ;

5.2 Numéro de carte (read only 32 bits)

Situé à l'adresse de base, permet de voir si la carte est bien présente
Base + 0 soit $(000000)_{bin}$ → $(00)_{hex}$

IDcode (32 bits)		
31	...	0

La valeur attendue à la lecture est $(DEAFBABE)_{hex}$.

5.3 Mise en mode acquisition (write 32 bits)

L'écriture dans ce registre valide les diverses voies. **Attention**, cette écriture doit être la dernière avant l'acquisition. Si la configuration doit être changée, il faut nécessairement désactiver les voies avant la reconfiguration.
Base + 2 soit $(000010)_{bin}$ (soit pour sur 32 bits base + 4) → $(04)_{hex}$

Voie 3	Voie 2	Voie 1	Voie 0
Bit 3	Bit 2	Bit 1	Bit 0

Voie activée si bit = 1.

5.4 DAC (write only 32 bits)

Il faut écrire à cette adresse les paramètres requis pour configurer les DAC, et donc pour positionner le piédestal.
Base + 4 + numéro de quart

- $(000100)_{bin}$: canal 1 et 2 ; (soit pour sur 32 bits base + 8) → $(08)_{hex}$
- $(000110)_{bin}$: canal 3 et 4 ; (soit pour sur 32 bits base + 12) → $(0C)_{hex}$

valeur (12 bits)			ADD (2 bits)	
29	…	18	17	16

Sur la carte, il y a physiquement 2 composants DAC implantés, ceux-ci à leur tour contiennent 4 sous-dac. C'est pour cela que pour adresser un DAC il y a une adresse VME $(08)_{hex}$ ou $(0C)_{hex}$ et une sous adresse contenue dans la donnée.
Par exemple, pour programmer les 2 DAC de la voie 0, il faut utiliser l'adresse $(08)_{hex}$ et faire successivement 2 écritures avec ADD à « 00 » et « 01 ».
Mapping des DAC :

Voie	Adresse	Registre DAC (17..16)
Ref0-	Base+8	00
Ref0+		01
Ref1-		10
Ref1+		11
Ref2-	Base+12	00
Ref2+		01
Ref3-		10
Ref3+		11

Tableau 6 : Adressage des DAC

Voici quelques exemples de données à transférer suivant le DAC désiré :

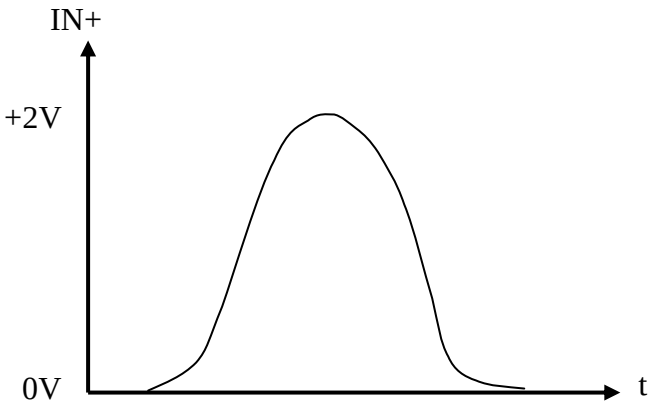
		REF0- REF2-	REF0+ REF2+	REF1- REF3-	REF1+ REF3+	
tension DAC	valeur DAC	sous-DAC=0	sous-DAC=1	sous-DAC=2	sous-DAC=3	tension X2
-2,50	0	0000	0001	0002	0003	-5
-2,40	163	028C	028D	028E	028F	-4,8
-2,30	327	051C	051D	051E	051F	-4,6
-2,20	491	07AC	07AD	07AE	07AF	-4,4
-2,10	655	0A3C	0A3D	0A3E	0A3F	-4,2
-2,00	819	0CCC	0CCD	0CCE	0CCF	-4
-1,90	983	0F5C	0F5D	0F5E	0F5F	-3,8
-1,80	1146	11E8	11E9	11EA	11EB	-3,6
-1,70	1310	1478	1479	147A	147B	-3,4
-1,60	1474	1708	1709	170A	170B	-3,2
-1,50	1638	1998	1999	199A	199B	-3
-1,40	1802	1C28	1C29	1C2A	1C2B	-2,8
-1,30	1966	1EB8	1EB9	1EBA	1EBB	-2,6
-1,20	2129	2144	2145	2146	2147	-2,4
-1,10	2293	23D4	23D5	23D6	23D7	-2,2
-1,00	2457	2664	2665	2666	2667	-2
-0,90	2621	28F4	28F5	28F6	28F7	-1,8
-0,80	2785	2B84	2B85	2B86	2B87	-1,6
-0,70	2949	2E14	2E15	2E16	2E17	-1,4
-0,60	3112	30A0	30A1	30A2	30A3	-1,2
-0,50	3276	3330	3331	3332	3333	-1
-0,40	3440	35C0	35C1	35C2	35C3	-0,8
-0,30	3604	3850	3851	3852	3853	-0,6
-0,20	3768	3AE0	3AE1	3AE2	3AE3	-0,4
-0,10	3932	3D70	3D71	3D72	3D73	-0,2
0,00	4096	4000	4001	4002	4003	0

Tableau 7 : tableau de valeurs DAC

Il faut charger une valeur différente dans les DAC en fonction des modes d'acquisition désirés. L'équation du DAC étant $Vout = 2 \times \left(\frac{2{,}5 \bullet N}{4096} - 2{,}5 \right)$ soit $N = \frac{(Vout/2 + 2.5) \bullet 4096}{2{,}5}$

5.4.1 Modes unipolaires

5.4.1.1 Unipolaire positif (0→2V)

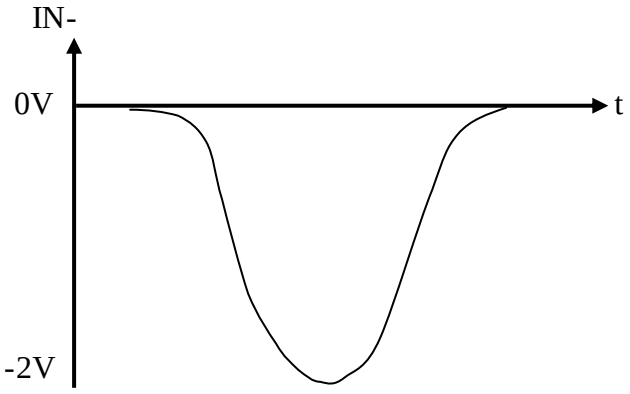


Le signal est connecté sur la voie IN+, un bouchon est positionné sur IN-. Il faut un offset -2,8V sur l'entrée positive et -3,8V sur l'entrée négative.
Ainsi, le signal sur :

- L'entrée négative de l'ADC va rester à 3,8V ;
- L'entrée négative de l'ADC va varier de 4,8V → 2,8V ;

Donc la différence à l'entrée de l'ADC va varier de -1V à +1V.

5.4.1.2 Unipolaire négatif (0→-2V)

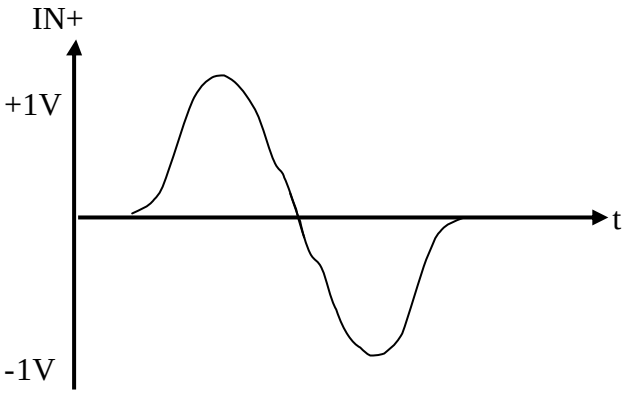


Le signal est connecté sur la voie IN-, un bouchon est positionné sur IN+. Il faut un offset -3,8V sur l'entrée positive et -4,8V sur l'entrée négative.
Ainsi, le signal sur :

- L'entrée négative de l'ADC va rester à 3,8V ;
- L'entrée négative de l'ADC va varier de 4,8V → 2,8V ;

Donc la différence à l'entrée de l'ADC va varier de -1V à +1V.

5.4.1.3 Bipolaire (-1V→1V)



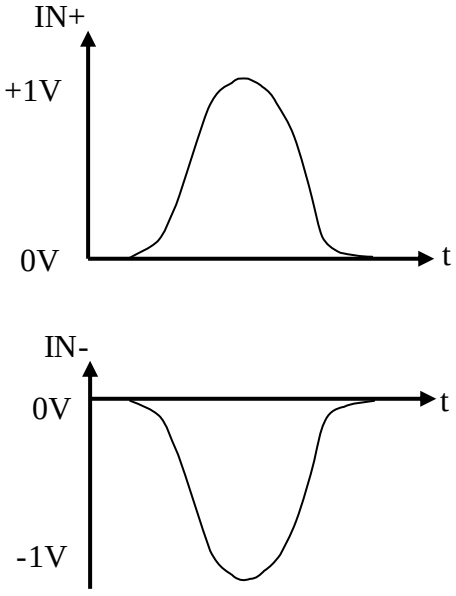
Le signal est connecté sur IN+, un bouchon est positionné sur IN-. Il faut un offset -3,8V sur l'entrée positive et -3,8V sur l'entrée négative.
Ainsi, le signal sur :

- L'entrée positive de l'ADC va varier de 2,8V → 4,8V ;
- L'entrée négative de l'ADC va rester à 3,8V ;

Donc la différence à l'entrée de l'ADC va varier de -1V à +1V.

5.4.2 Modes différentiels

5.4.2.1 Différentiel unipolaire (0→1V) et (0V→-1V)

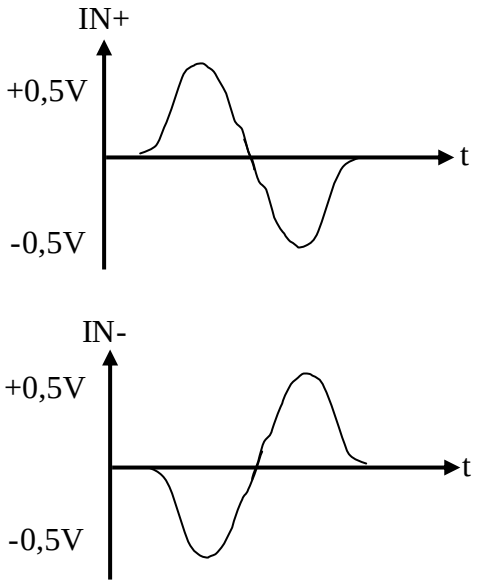


Les signaux sont connectés sur IN- et IN+. Il faut un offset -3,5V sur l'entrée positive et -4,5V sur l'entrée négative.
Ainsi, le signal sur :

- L'entrée positive de l'ADC va varier de 3,5V → 4,5V ;
- L'entrée négative de l'ADC va varier de 4,5V → 3,5V ;

Donc la différence à l'entrée de l'ADC va varier de -1V à +1V.

5.4.2.2 Différentiel bipolaire (-0,5V→0,5V) et (-0,5V→0,5V)



Les signaux sont connectés sur IN- et IN+. Il faut un offset -4V sur l'entrée positive et -4V sur l'entrée négative.
Ainsi, le signal sur :

- L'entrée positive de l'ADC va varier de 3,5V → 4,5V ;
- L'entrée négative de l'ADC va varier de 4,5V → 3,5V ;

Donc la différence à l'entrée de l'ADC va varier de -1V à +1V.

5.5 Largeur fenêtre, seuil et n (write only 32 bits)

Une écriture à cette adresse permet de configurer la largeur de fenêtre, le seuil à franchir par le pic et le nombre de données à conserver après la sortie du pic.
Base + 8 + n° canal

- $(001000)_{bin}$: canal 0 ; (soit pour sur 32 bits base + 16) → $(10)_{hex}$
- $(001010)_{bin}$: canal 1 ; (soit pour sur 32 bits base + 20) → $(14)_{hex}$
- $(001100)_{bin}$: canal 2 ; (soit pour sur 32 bits base + 24) → $(18)_{hex}$
- $(001110)_{bin}$: canal 3 ; (soit pour sur 32 bits base + 28) → $(1C)_{hex}$

½ Largeur_fenetre (14 bits)			Seuil (10 bits)			N (8 bits)		
31	…	18	17	…	8	7	…	0

Attention : N doit toujours être strictement supérieur au nombre de données avant pic, la largeur de fenêtre min doit être de 10 ! La largeur de fenêtre (pas la demi) doit de préférence être supérieure au nombre de données avant stop !

5.6 Nombre de données avant stop (write only 32 bits)

Une écriture à cette adresse permet de configurer le seuil *almost empty* de la FIFO brute, et donc le nombre de donnée avant le stop.
Base + 16 + n° canal

- $(010000)_{bin}$: canal 0 ; (soit pour sur 32 bits base + 32) → $(20)_{hex}$
- $(010010)_{bin}$: canal 1 ; (soit pour sur 32 bits base + 36) → $(24)_{hex}$
- $(010100)_{bin}$: canal 2 ; (soit pour sur 32 bits base + 40) → $(28)_{hex}$
- $(010110)_{bin}$: canal 3 ; (soit pour sur 32 bits base + 44) → $(2C)_{hex}$

PAF : doit être programmé à 32768-(val_AF-2), ou val_AF est le nombre de données avant stop. **Attention,** il faut val_AF>3 ! PAE peut être nul.

Non utilisé		PAF (15 bits)			PAE (15 bits)		
31	30	29	…	15	14	…	0

5.7 Nombre de données avant pic (M) (write only 32 bits)

Une écriture à cette adresse permet de configurer le seuil *almost empty* de la FIFO annexe, et donc le nombre de donnée avant le franchissement de seuil.
Base + 24 + n° canal

- $(011000)_{bin}$: canal 0 ; (soit pour sur 32 bits base + 48) → $(30)_{hex}$
- $(011010)_{bin}$: canal 1 ; (soit pour sur 32 bits base + 52) → $(34)_{hex}$
- $(011100)_{bin}$: canal 2 ; (soit pour sur 32 bits base + 56) → $(38)_{hex}$
- $(011110)_{bin}$: canal 3 ; (soit pour sur 32 bits base + 60) → $(3C)_{hex}$

PAF : doit être programmé à 2048-(val_AF-2), ou val_AF est le nombre de données avant le FIFO pleine. **Attention,** il faut val_AF>3 ! PAE peut être nul. Cette valeur doit être strictement inférieure à N.

Non utilisé			PAF (11 bits)			PAE (11 bits)		
31	…	22	21	…	11	10	…	0

5.8 Registre de sortie (read only 32 bits)

Permet de lire les données de la voie sélectionnée.
Base + 32 + n° canal

- $(100000)_{bin}$: canal 0 ; (soit pour sur 32 bits base + 64) → $(40)_{hex}$
- $(100010)_{bin}$: canal 1 ; (soit pour sur 32 bits base + 68) → $(44)_{hex}$
- $(100100)_{bin}$: canal 2 ; (soit pour sur 32 bits base + 72) → $(48)_{hex}$
- $(100110)_{bin}$: canal 3 ; (soit pour sur 32 bits base + 76) → $(4C)_{hex}$

Pour récupérer les données il faudrait récupérer les données canal après canal en utilisant au mieux possible la bande passante. Il y a 2 types de données pouvant être issues de la FIFO :

- un mot de contrôle (marque relative par exemple) ;
- des données ADC ;

Il faut donc un différenciateur entre ces 2 données . 1 bit suffit.

Pour utiliser tous les bits du bus de données 32 bits, il faut faire attention à la disposition des données. Dans tous les cas, il y aura un transfert de 30 bits de données et de 2 bits d'identification du mot. Les bits d'identification :

- 00 : marqueur de pic relatif ;
- 01 : données ;
- 10 : fin de pic (nombre de données transmises) ;
- 11 : fin de fenêtre (bits(29..0)=0000 si fin de fenêtre et bits(29..0)=FFFF si FIFO vide) ;

Data [31..30]	Data [29..0]
00	Position relative du pic (15 bits)
01	Données (30 bits)
01	
10	fin de pic (15 bits)
00	Position relative du pic (15 bits)
01	Données (30 bits)
01	
10	fin de pic (15 bits)
. . .	. . .
11	Fin de fenêtre

Le rangement des données est tel que pour trouver le nombre de mots utiles dans la dernière lecture de données, il suffit de faire une opération nombre de données transmises modulo 3, et suivant le résultat :

- 0 : on garde les trois données ;
- 1 : on garde 1 donnée ;
- 2 : on garde 2 données ;

Exemples :

- si il y a 4 données : 4 mod 3 =1

Donnée1	Donnée2	Donnée3
XXXXX	XXXXX	Donnée4

- si il y a 5 données : 5 mod 3 =2

Donnée1	Donnée2	Donnée3
XXXXX	Donnée4	Donnée5

- si il y a 6 données : 6 mod 3 =0

Donnée1	Donnée2	Donnée3
Donnée4	Donnée5	Donnée6

Donc pour chaque stop valide, une marque de fin de fenêtre est insérée. Ainsi, lorsque l'on est en fonctionnement 2 modes de fonctionnement restent possible :

- IT sur FIFO non vide, (une IT pour tout trigger accepté, hors temps mort). Sur cette IT lire toutes les voies jusqu'à tomber sur la marque de fin de fenêtre.
- IT si le nombre de données dans FIFO supérieur à un seuil ou si un pic est détecté, Le nombre d'IT est conditionné par le nombre d'événements sur voie la plus rapide. Inconvénient, latence plus grande. Dans le status d'IT il faut indiquer si IT sur seuil ou

sur pic. Si sur seuil, le nombre minimum de données à lire (= au pointeur) sur chaque voie est connu.

Pour garder le synchronisme entre les voies, elles doivent être configurées de la même manière. Si le synchronisme n'importe pas, les voies deviennent totalement indépendantes, et la technique de balayage doit être revue en conséquence.

Bien respecter ce qui est écrit. Une façon de faire la navigation dans les diverses branches (MSB=00,01,10,11) est de faire le test en 2 étapes, tester nombre positif ou non, puis décaler à gauche de 2 et refaire le test. **Attention**, cela est valable uniquement si le soft manipule des integer 32 bits.

5.9 Nombre d'événements rejetés (read only 32 bits)

Sert à relire le compteur de la voie sélectionnée. Pour effacer ce compteur, il faut « reseter » la carte.
Base + 40 + n° canal

- $(101000)_{bin}$: canal 0 ; (soit pour sur 32 bits base + 80) → $(50)_{hex}$
- $(101010)_{bin}$: canal 1 ; (soit pour sur 32 bits base + 84) → $(54)_{hex}$
- $(101100)_{bin}$: canal 2 ; (soit pour sur 32 bits base + 88) → $(58)_{hex}$
- $(101110)_{bin}$: canal 3 ; (soit pour sur 32 bits base + 92) → $(5C)_{hex}$

Nb événement rejetés (14 bits)			Non utilisés			Non utilisés		
31	…	18	17	…	8	7	…	0

5.10 Configuration interruption (write 32 bits)

L'écriture dans ce registre valide les diverses interruptions. (2 différentes en vert et en bleu).
Base + 48 : $(110000)_{bin}$; (soit pour sur 32 bits base + 96) → $(60)_{hex}$

Prog_IT_ot			Prog_IT_FF			Val_AF	Val_AE	Val_EF	Val_FF
9	8	7	6	5	4	3	2	1	0

Val_FF : valide l'IT si une FIFO est vue pleine
Val_EF : valide l'IT si une FIFO est vue non vide
Val_AE : valide l'IT si une FIFO n'est pas presque vide (nb data >1023)
Val_AF : valide l'IT si une FIFO est vue presque pleine (nb data >31k)
Prog_IT_FF : donne le niveau de priorité de l'IT FF (sans teinte)
Prog_IT_ot : donne le niveau de priorité des IT EF, PAE et PAF (en bleu)

Les 2 valeurs peuvent être différentes. Elles doivent être positionnées à zéro si non utilisées.
L'IT FF est prioritaire (à niveau d'IRQ égal) par rapport aux autres IRQ.
Dans le cas d'une interruption, il faut lire les vecteurs, puis régler les sources de problèmes en revalidant les IT, et enfin il faut refaire une écriture dans ce registre pour réactiver les IT. Attention, toutes les IT se réactivent en même temps. Il faut donc soit traiter toutes les sources de problèmes. Soit réactiver une IT à la fois.
Les vecteurs d'interruption retournés sont :

- Pour FIFO vide : $(F0)_{hex}$;
- Pour FIFO presque vide : $(F2)_{hex}$;

- Pour FIFO presque pleine : $(F4)_{hex}$;
- Pour FIFO pleine : $(F8)_{hex}$;

5.11 Registre de reset soft (write 32 bits)

L'écriture d'un '1' dans ce registre met les voies d'acquisition en reset (pas l'EPLD d'interface VME). Le reset doit durer au moins 1 ms. **Attention**, avant de reconfigurer la carte, il désactiver les voies (car cela est contrôlé par l'EPLD VME).
Base + 56 : $(111000)_{bin}$ (soit pour sur 32 bits base + 112) → $(70)_{hex}$

Après avoir reseté la carte attendre au moins 50 ms pour la reconfigurer.

5.12 Récapitulatif

Adresse	registre	mode
Base +$(00)_{hex}$	IDCODE	READ 32
Base +$(04)_{hex}$	Mise en mode acquisition	WRITE 32
Base +$(08)_{hex}$	Programmation DAC voie 0 et 1	WRITE 32
Base +$(0C)_{hex}$	Programmation DAC voie 2 et 3	WRITE 32
Base +$(10)_{hex}$	Largeur fenêtre, seuil et n de la voie 0	WRITE 32
Base +$(14)_{hex}$	Largeur fenêtre, seuil et n de la voie 1	WRITE 32
Base +$(18)_{hex}$	Largeur fenêtre, seuil et n de la voie 2	WRITE 32
Base +$(1C)_{hex}$	Largeur fenêtre, seuil et n de la voie 3	WRITE 32
Base +$(20)_{hex}$	Nombre de données avant stop de la voie 0	WRITE 32
Base +$(24)_{hex}$	Nombre de données avant stop de la voie 1	WRITE 32
Base +$(28)_{hex}$	Nombre de données avant stop de la voie 2	WRITE 32
Base +$(2C)_{hex}$	Nombre de données avant stop de la voie 3	WRITE 32
Base +$(30)_{hex}$	Nombre de données avant pic de la voie 0	WRITE 32
Base +$(34)_{hex}$	Nombre de données avant pic de la voie 1	WRITE 32
Base +$(38)_{hex}$	Nombre de données avant pic de la voie 2	WRITE 32
Base +$(3C)_{hex}$	Nombre de données avant pic de la voie 3	WRITE 32
Base +$(40)_{hex}$	Registre de sortie de la voie 0	READ 32
Base +$(44)_{hex}$	Registre de sortie de la voie 1	READ 32
Base +$(48)_{hex}$	Registre de sortie de la voie 2	READ 32
Base +$(4C)_{hex}$	Registre de sortie de la voie 3	READ 32
Base +$(50)_{hex}$	Compteur d'événements rejetés de la voie 0	READ 32
Base +$(54)_{hex}$	Compteur d'événements rejetés de la voie 1	READ 32
Base +$(58)_{hex}$	Compteur d'événements rejetés de la voie 2	READ 32
Base +$(5C)_{hex}$	Compteur d'événements rejetés de la voie 3	READ 32
Base +$(60)_{hex}$	Configuration des interruptions	WRITE 32
Base +$(70)_{hex}$	Registre de reset	WRITE 32

Tableau 8 : Synthèse des adresses

6 Précautions software

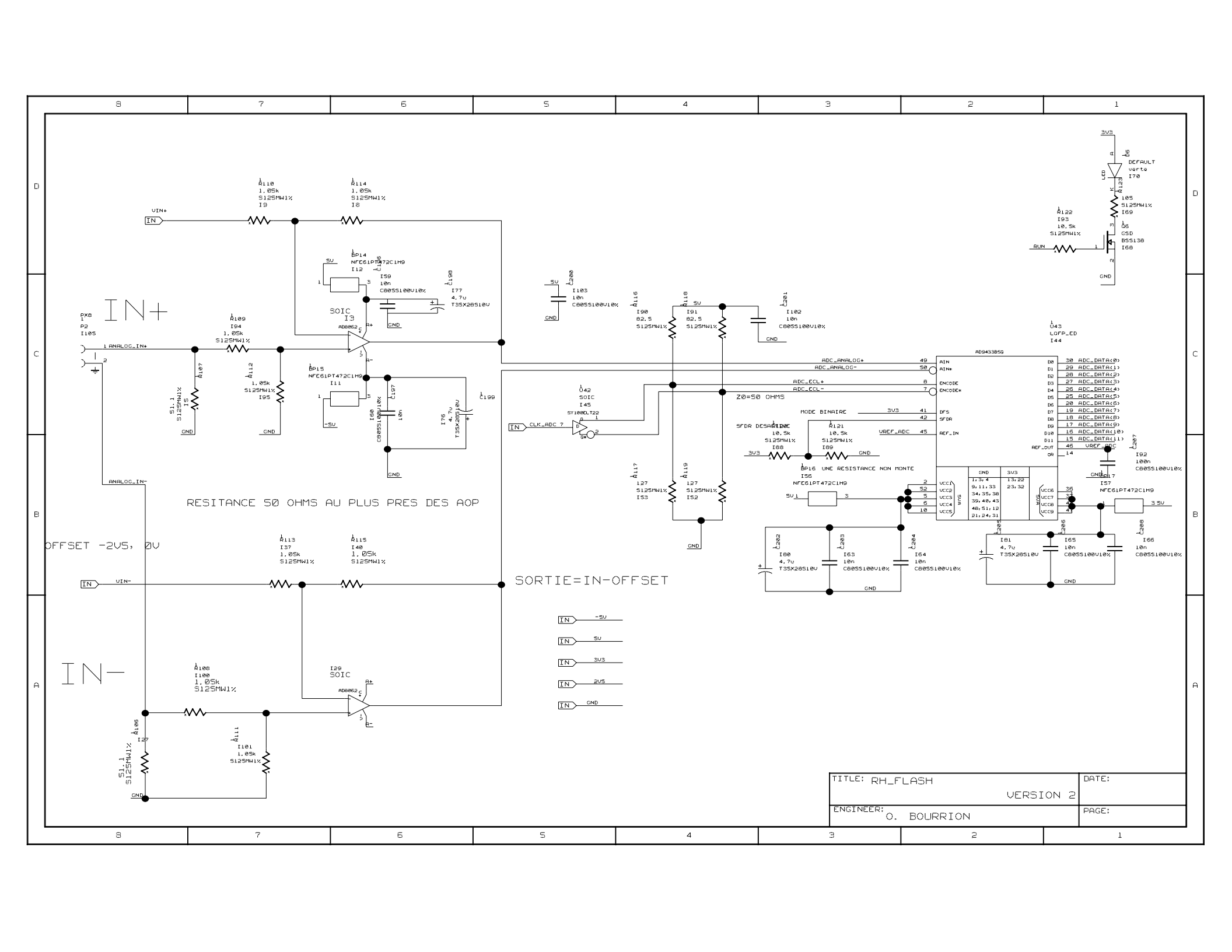
Toute la carte doit être configurée avant de la passer en mode run. Si la carte est « resetée », il ne faut pas oublier de reprendre la programmation de la carte depuis le début.

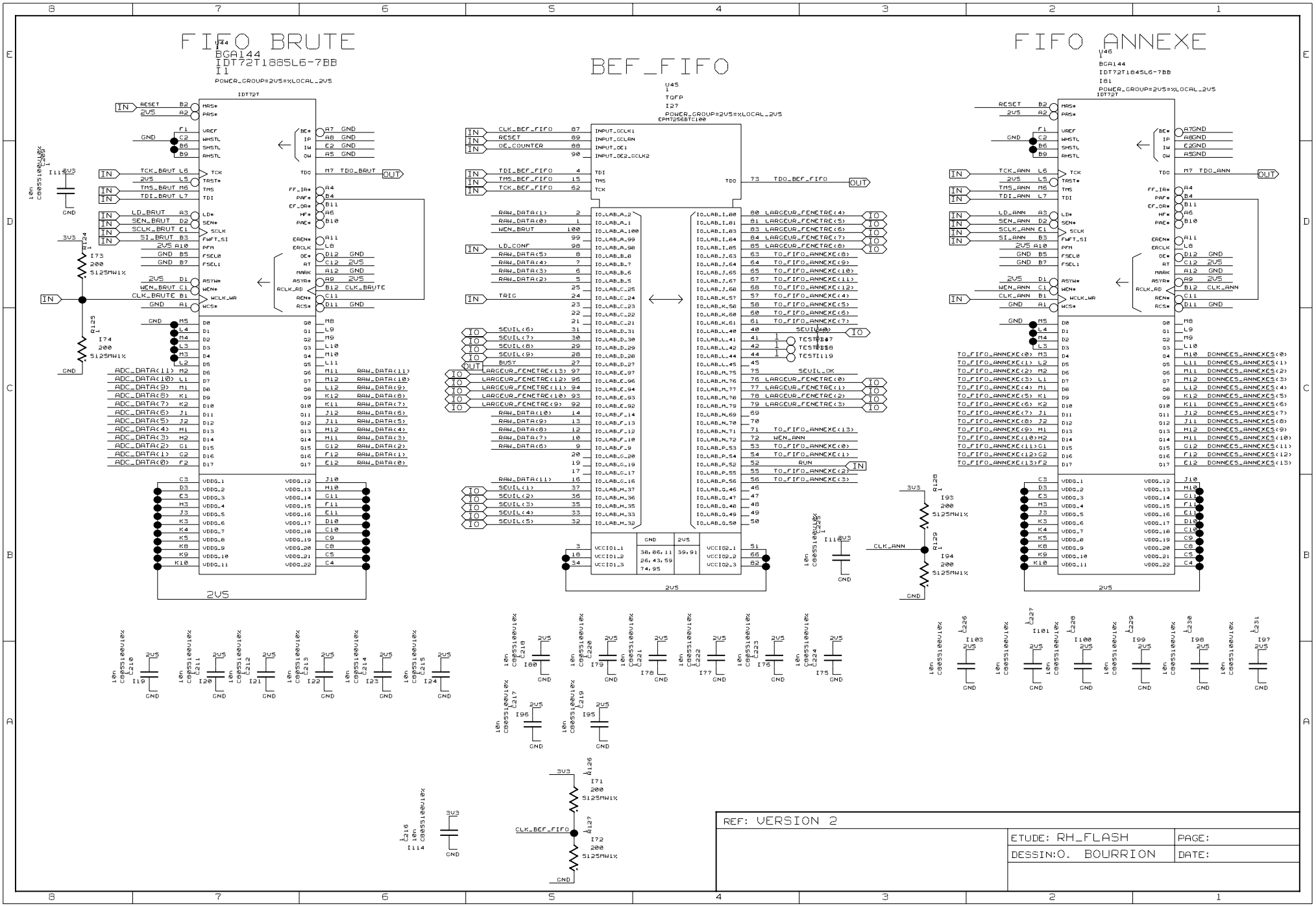
Lors de la programmation de m et n il faut toujours que N soit strictement supérieur à M, sinon dans le cas de pics trop proches, on risque de ne pas re-déclencher.

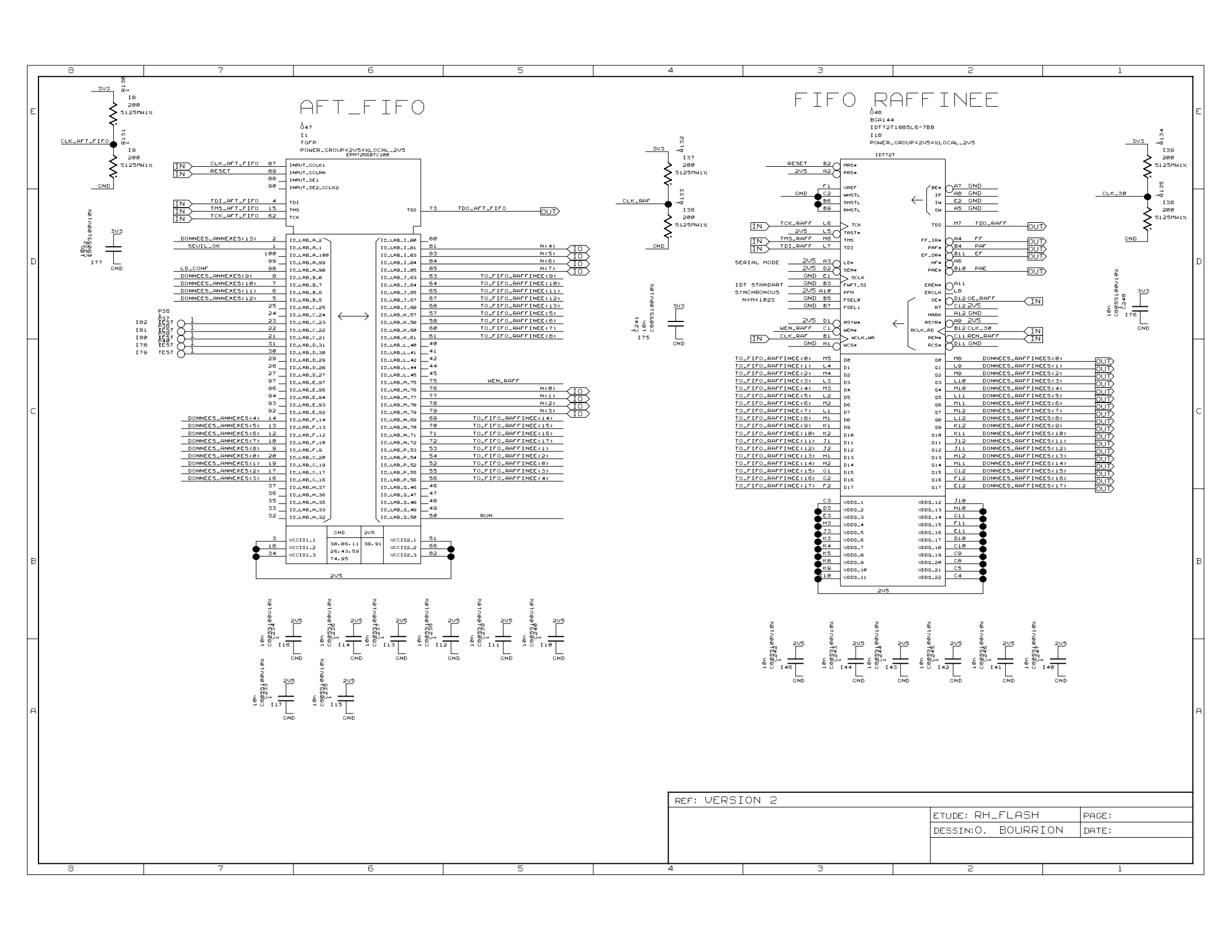
Les interruptions s'auto devalident lorsqu'elles se déclenchent. Il faut donc penser à les réarmer en écrivant dans le registre de masque.

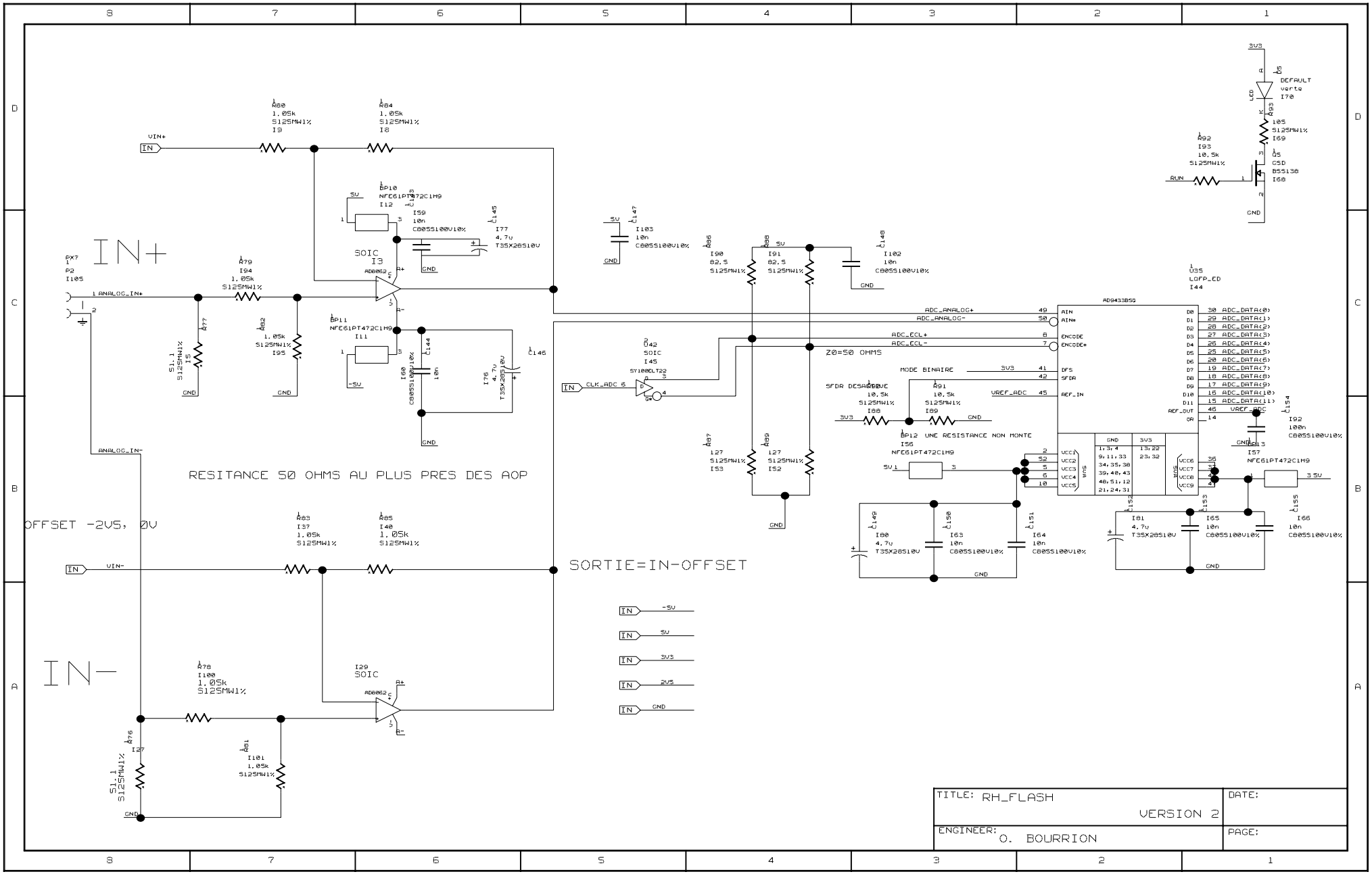
Si l'interruption FULL FIFO (FF) se déclenche, cela signifie que des données ont été perdues, il vaut mieux « reseter » la carte.

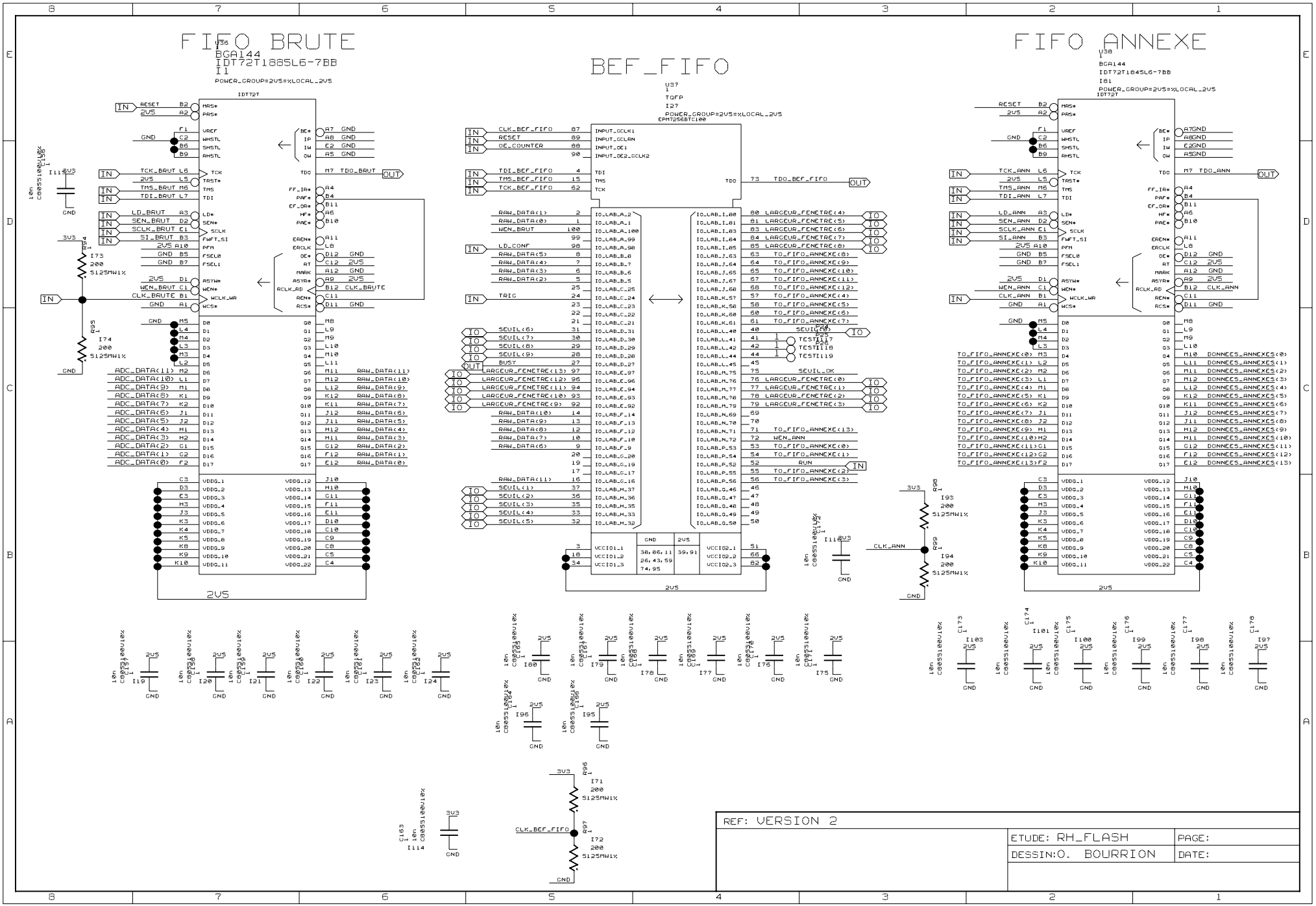
Si l'on veut compter les événements et garder les voies synchrones, il faut configurer une largeur de fenêtre et des données avant stop identiques sur toutes les voies. Dans le cas contraire certaines voies risquent d'être busy pendant que d'autres déclenchent.


IN+
IN-
OFFSET -2V5, 0V
RESITANCE 50 OHMS AU PLUS PRES DES AOP
SORTIE=IN-OFFSET
Z0=50 OHMS
UNE RESISTANCE NON MONTE
MODE BINAIRE
SOIC
VIN+
VIN-
ANALOG_IN+
ANALOG_IN-
CLK_ADC
ADC_ANALOG+
ADC_ANALOG-
ADC_ECL+
ADC_ECL-
VREF_ADC
AD9433BSQ
LQFP_ED
DEFAULT
RUN
GND
3V3
5V
-5V
2V5
TITLE: RH_FLASH
VERSION 2
DATE:
ENGINEER: O. BOURRION
PAGE:

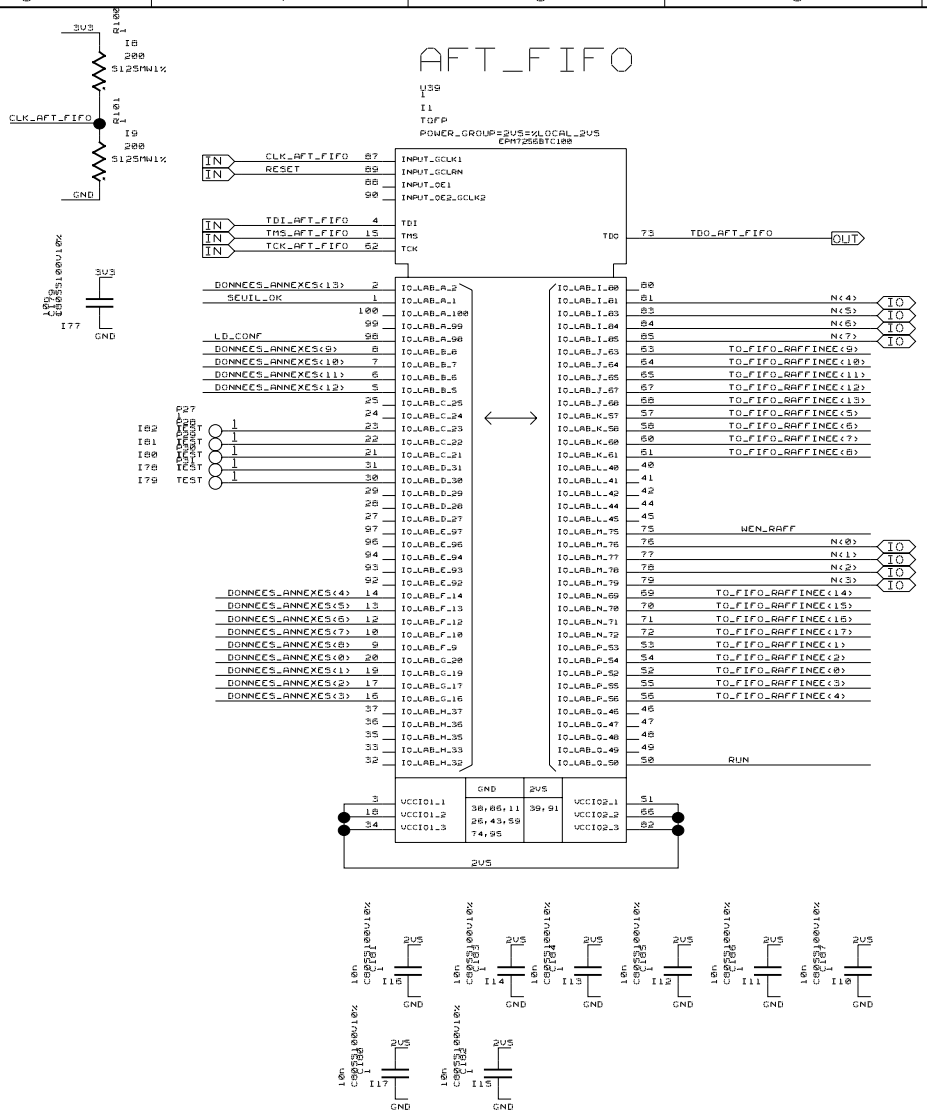

FIFO_BRUTE
U44
BGA144
IDT72T1885L6-7BB
I1
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
BEF_FIFO
U45
1
TQFP
I27
POWER_GROUP=2V5=%LOCAL_2V5
EPM7256BTC100
FIFO_ANNEXE
U46
BGA144
IDT72T1845L6-7BB
I81
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
CLK_BEF_FIFO
CLK_ANN
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:


AFT_FIFO
U47
I1
TQFP
POWER_GROUP=2V5=%LOCAL_2V5
CPM7256BTC100
FIFO_RAFFINEE
U48
BGA144
IDT72T1885L6-7BB
I18
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
CLK_AFT_FIFO
RESET
TDI_AFT_FIFO
TMS_AFT_FIFO
TCK_AFT_FIFO
TDO_AFT_FIFO
CLK_RAF
TCK_RAFF
TMS_RAFF
TDI_RAFF
TDO_RAFF
CLK_30
WEN_RAFF
REN_RAFF
SERIAL MODE
IDT STANDART
SYNCHRONOUS
N=M=1023
LD_CONF
SEUIL_OK
TEST
RUN
DONNEES_ANNEXES(0)
DONNEES_ANNEXES(1)
DONNEES_ANNEXES(2)
DONNEES_ANNEXES(3)
DONNEES_ANNEXES(4)
DONNEES_ANNEXES(5)
DONNEES_ANNEXES(6)
DONNEES_ANNEXES(7)
DONNEES_ANNEXES(8)
DONNEES_ANNEXES(9)
DONNEES_ANNEXES(10)
DONNEES_ANNEXES(11)
DONNEES_ANNEXES(12)
DONNEES_ANNEXES(13)
N(0)
N(1)
N(2)
N(3)
N(4)
N(5)
N(6)
N(7)
TO_FIFO_RAFFINEE(0)
TO_FIFO_RAFFINEE(1)
TO_FIFO_RAFFINEE(2)
TO_FIFO_RAFFINEE(3)
TO_FIFO_RAFFINEE(4)
TO_FIFO_RAFFINEE(5)
TO_FIFO_RAFFINEE(6)
TO_FIFO_RAFFINEE(7)
TO_FIFO_RAFFINEE(8)
TO_FIFO_RAFFINEE(9)
TO_FIFO_RAFFINEE(10)
TO_FIFO_RAFFINEE(11)
TO_FIFO_RAFFINEE(12)
TO_FIFO_RAFFINEE(13)
TO_FIFO_RAFFINEE(14)
TO_FIFO_RAFFINEE(15)
TO_FIFO_RAFFINEE(16)
TO_FIFO_RAFFINEE(17)
DONNEES_RAFFINEES(0)
DONNEES_RAFFINEES(1)
DONNEES_RAFFINEES(2)
DONNEES_RAFFINEES(3)
DONNEES_RAFFINEES(4)
DONNEES_RAFFINEES(5)
DONNEES_RAFFINEES(6)
DONNEES_RAFFINEES(7)
DONNEES_RAFFINEES(8)
DONNEES_RAFFINEES(9)
DONNEES_RAFFINEES(10)
DONNEES_RAFFINEES(11)
DONNEES_RAFFINEES(12)
DONNEES_RAFFINEES(13)
DONNEES_RAFFINEES(14)
DONNEES_RAFFINEES(15)
DONNEES_RAFFINEES(16)
DONNEES_RAFFINEES(17)
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:

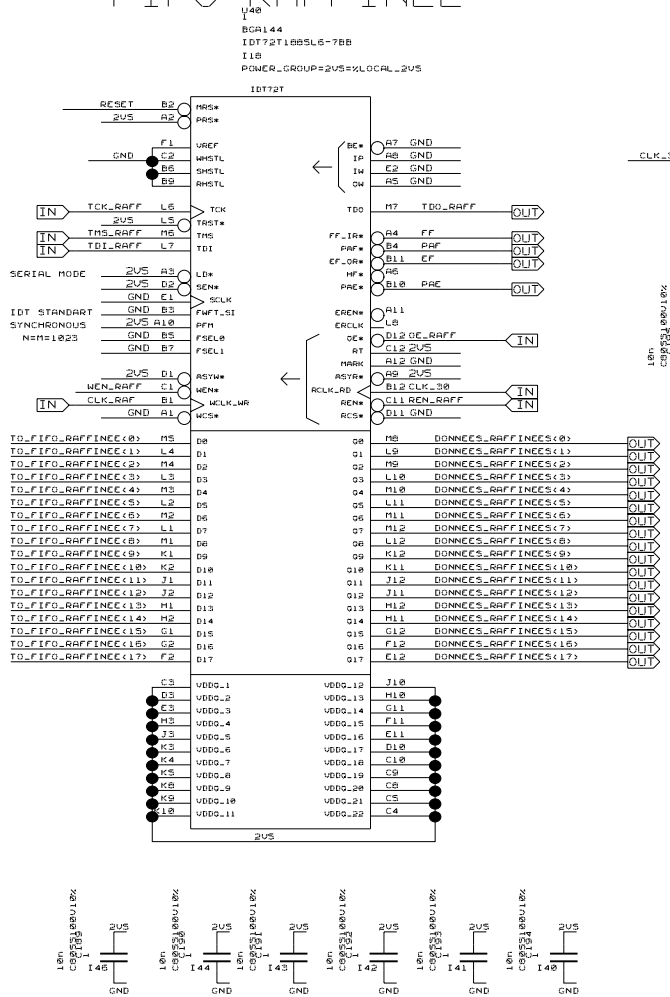

IN+
IN-
VIN+
VIN-
ANALOG_IN+
ANALOG_IN-
RESITANCE 50 OHMS AU PLUS PRES DES AOP
OFFSET -2V5, 0V
SORTIE=IN-OFFSET
SOIC
AD8062
NFE61PT472C1H9
ADC_ANALOG+
ADC_ANALOG-
ADC_ECL+
ADC_ECL-
Z0=50 OHMS
CLK_ADC 6
MODE BINAIRE
VREF_ADC
UNE RESISTANCE NON MONTE
AD9433BSQ
LQFP_ED
DEFAULT
verte
BSS138
RUN
-5V
5V
3V3
2V5
GND
TITLE: RH_FLASH
VERSION 2
ENGINEER: O. BOURRION
DATE:
PAGE:


FIFO_BRUTE
BEF_FIFO
FIFO_ANNEXE
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:

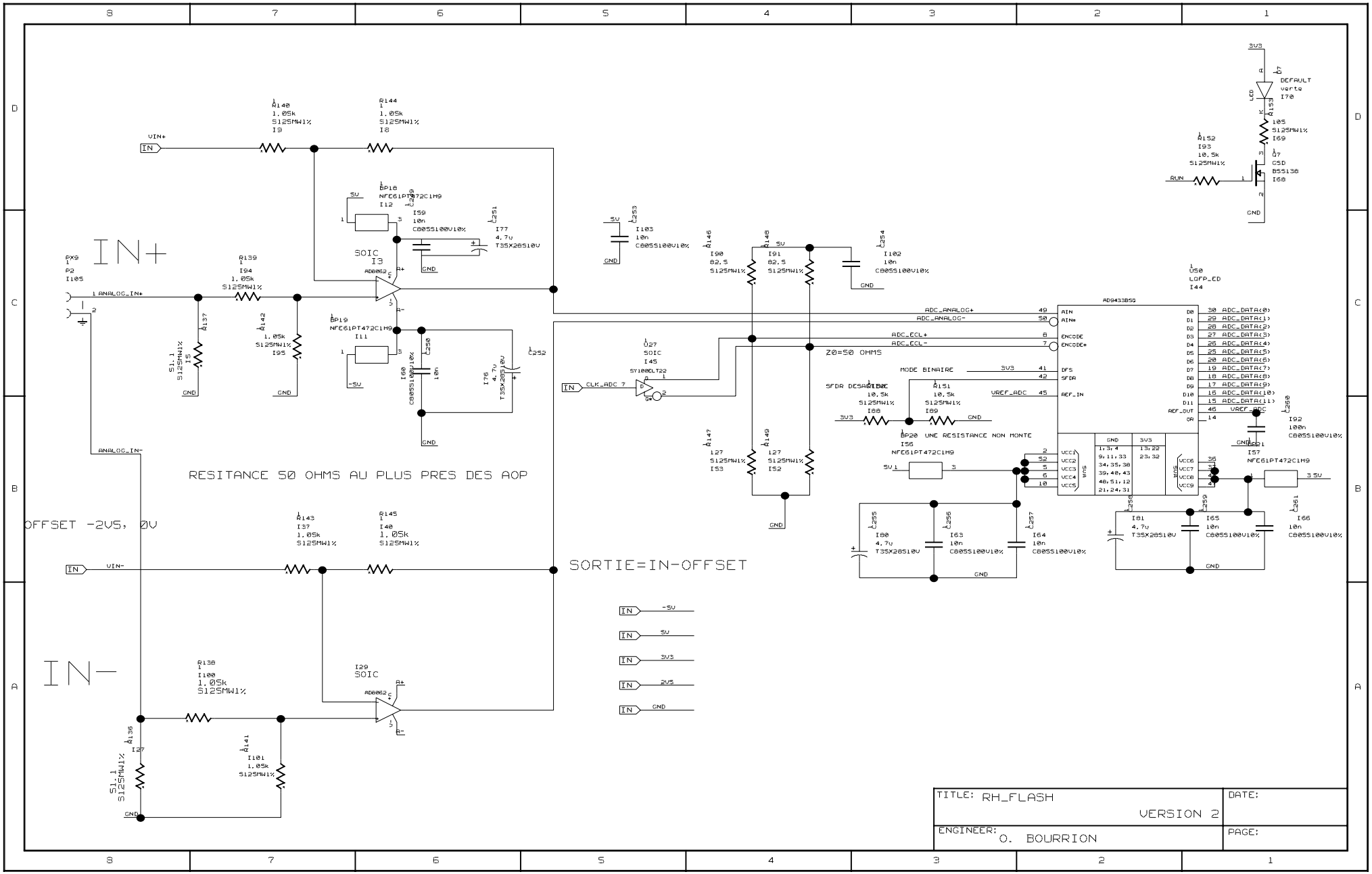
AFT_FIFO

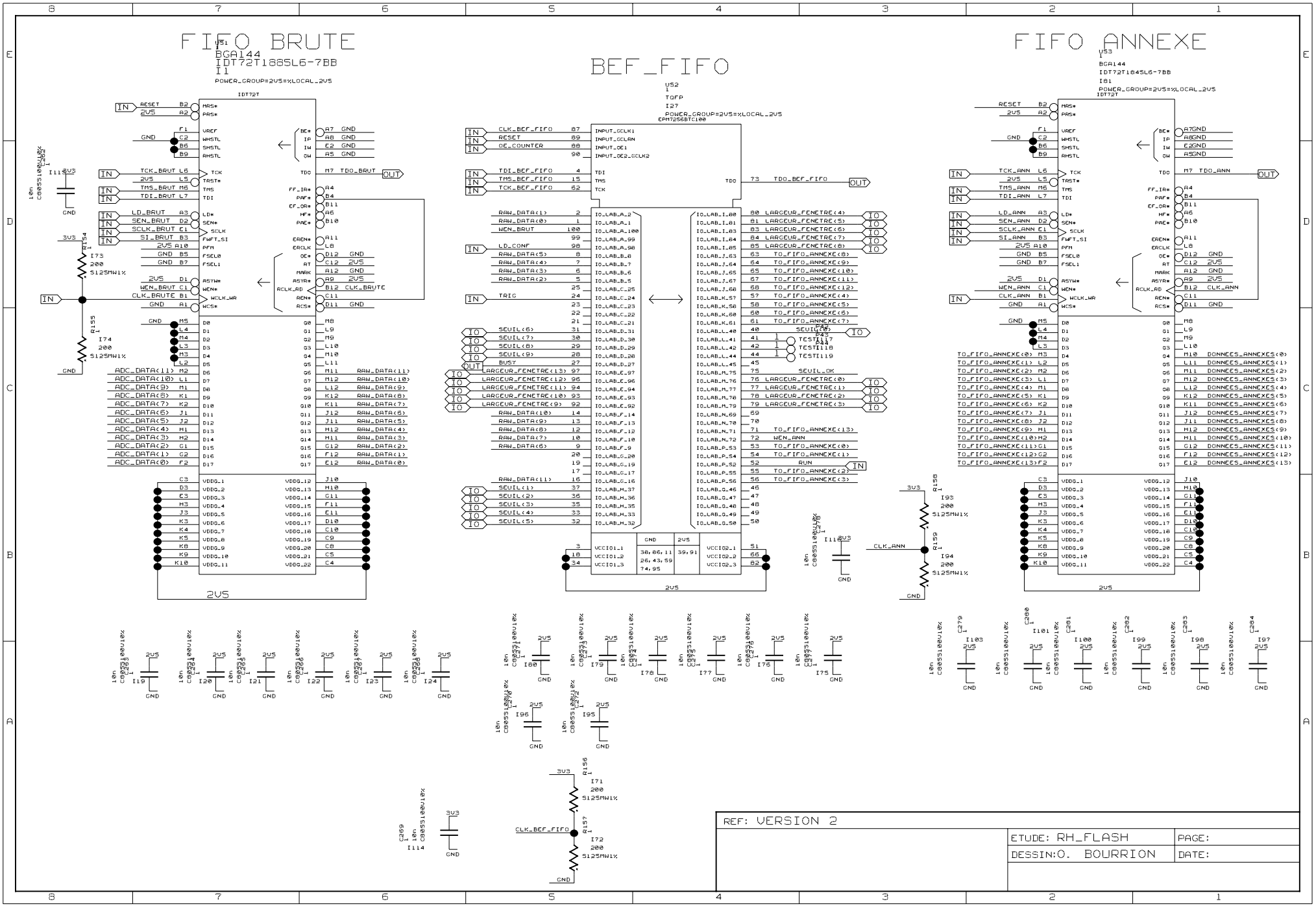


FIFO_RAFFINEE

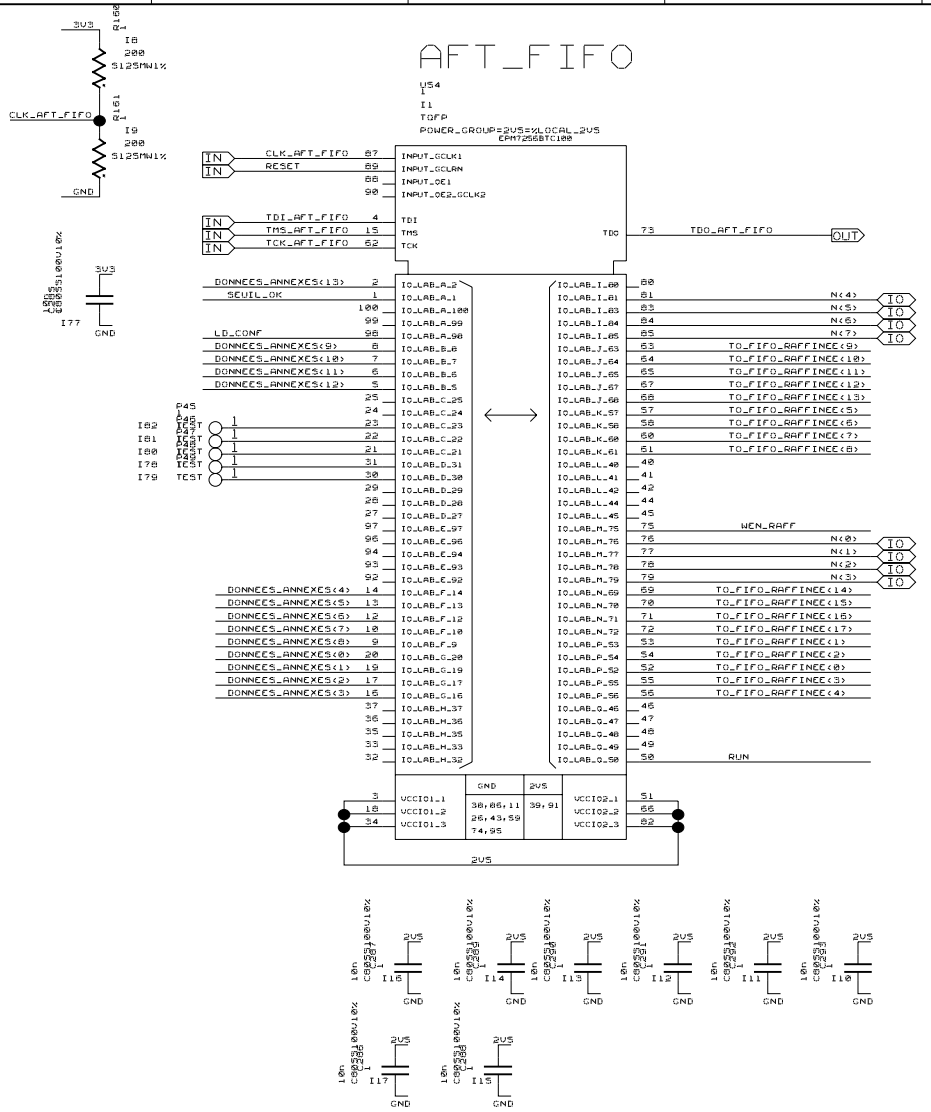


REF: VERSION 2		
	ETUDE: RH_FLASH	PAGE:
	DESSIN:O. BOURRION	DATE:

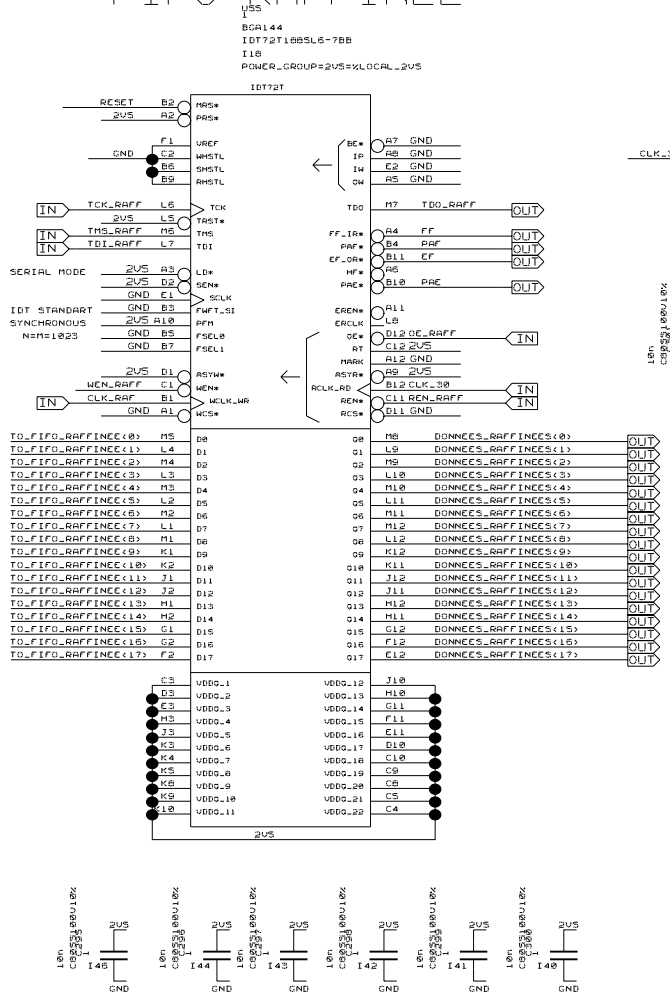

IN+
IN-
OFFSET -2V5, 0V
SORTIE=IN-OFFSET
RESITANCE 50 OHMS AU PLUS PRES DES AOP
Z0=50 OHMS
UNE RESISTANCE NON MONTE
MODE BINAIRE
VIN+
VIN-
ANALOG_IN+
ANALOG_IN-
CLK_ADC 7
ADC_ANALOG+
ADC_ANALOG-
ADC_ECL+
ADC_ECL-
VREF_ADC
RUN
DEFAULT
verte
AD9433BSQ
LQFP_ED
SOIC
GND
3V3
5V
-5V
2V5
TITLE: RH_FLASH
VERSION 2
ENGINEER: O. BOURRION
DATE:
PAGE:


FIFO_BRUTE
U51
BGA144
IDT72T1885L6-7BB
I1
POWER_GROUP=2V5=%LOCAL_2V5
BEF_FIFO
U52
TQFP
I27
POWER_GROUP=2V5=%LOCAL_2V5
EPM7256BTC100
FIFO_ANNEXE
U53
BGA144
IDT72T1845L6-7BB
I81
POWER_GROUP=2V5=%LOCAL_2V5
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:

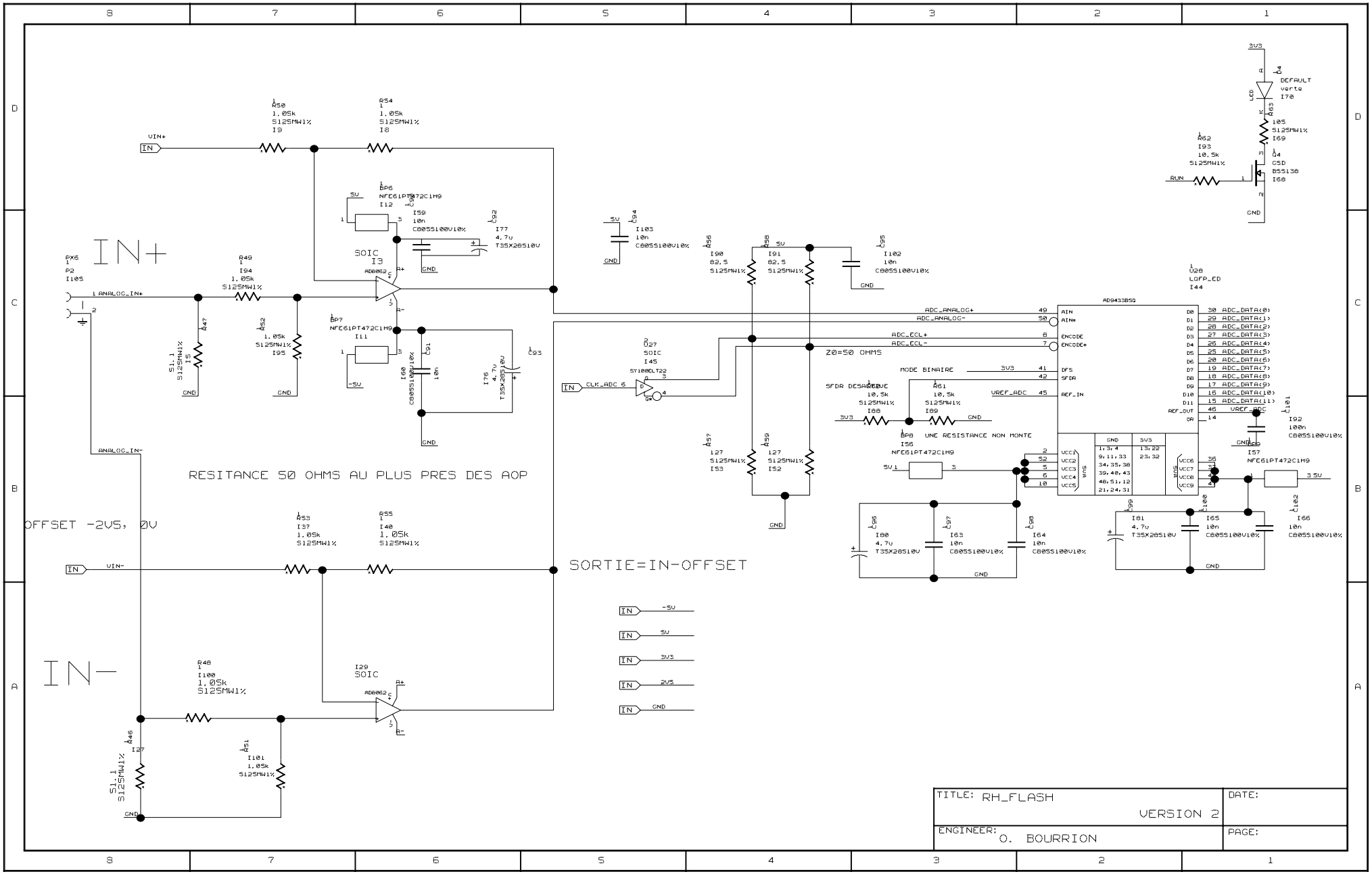
AFT_FIFO

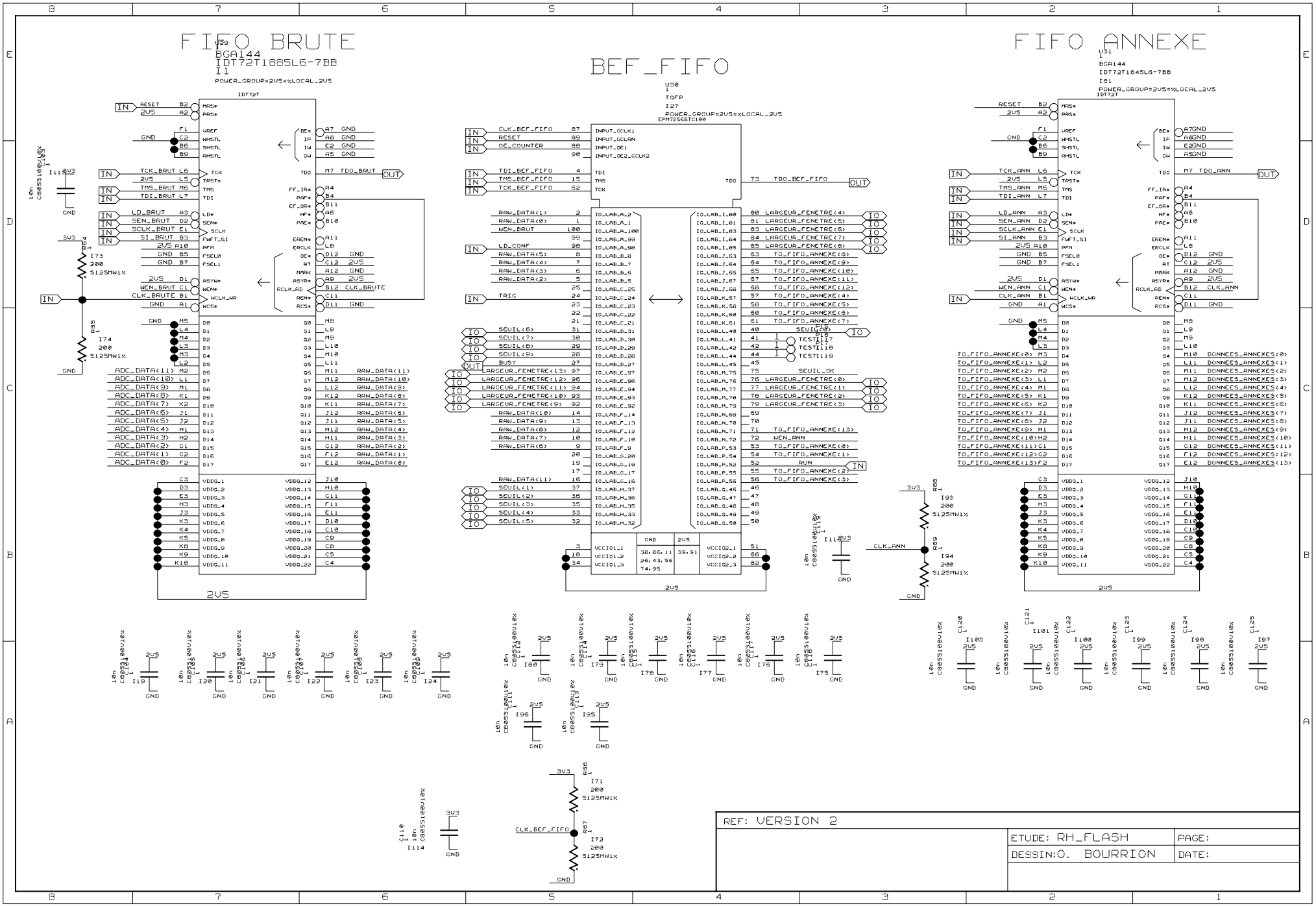


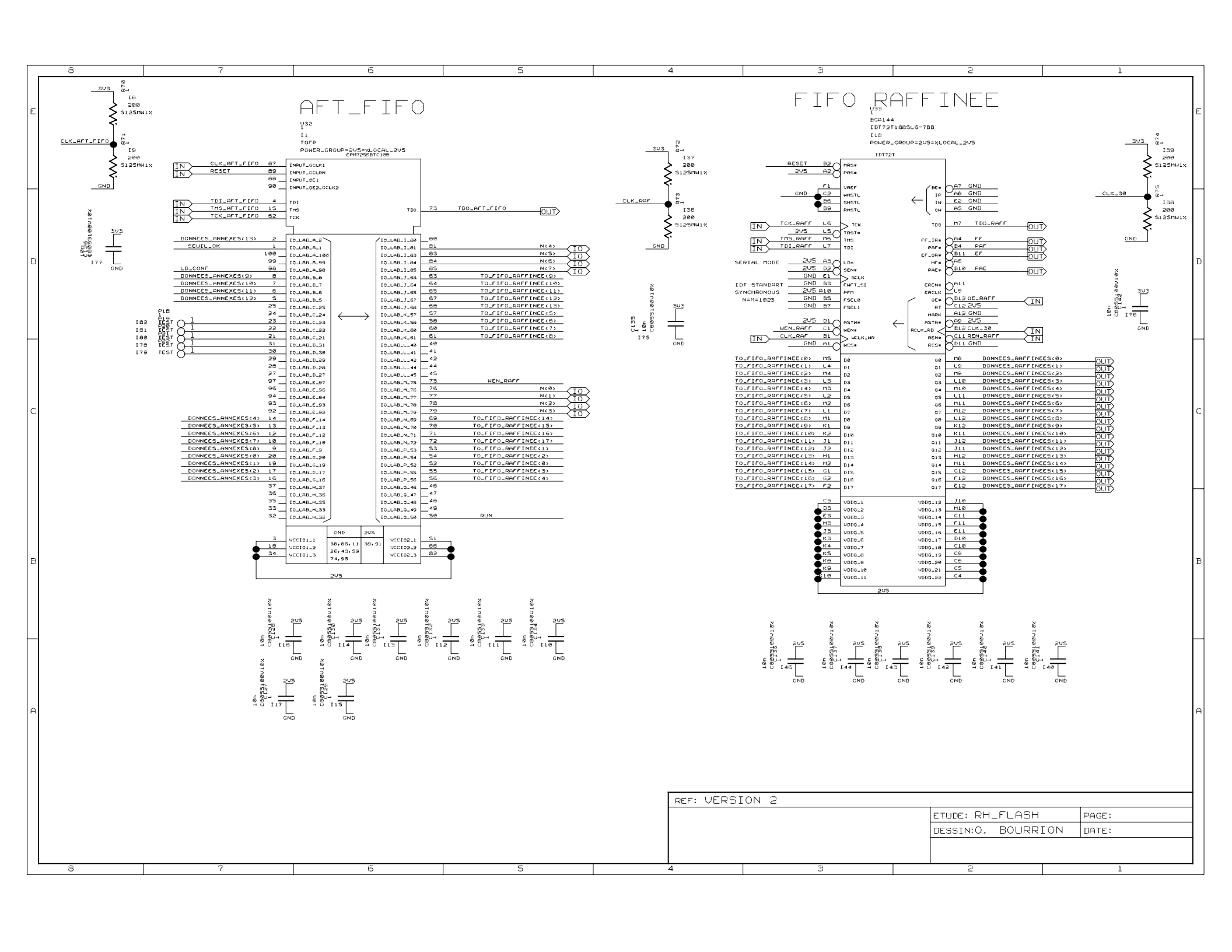
FIFO_RAFFINEE

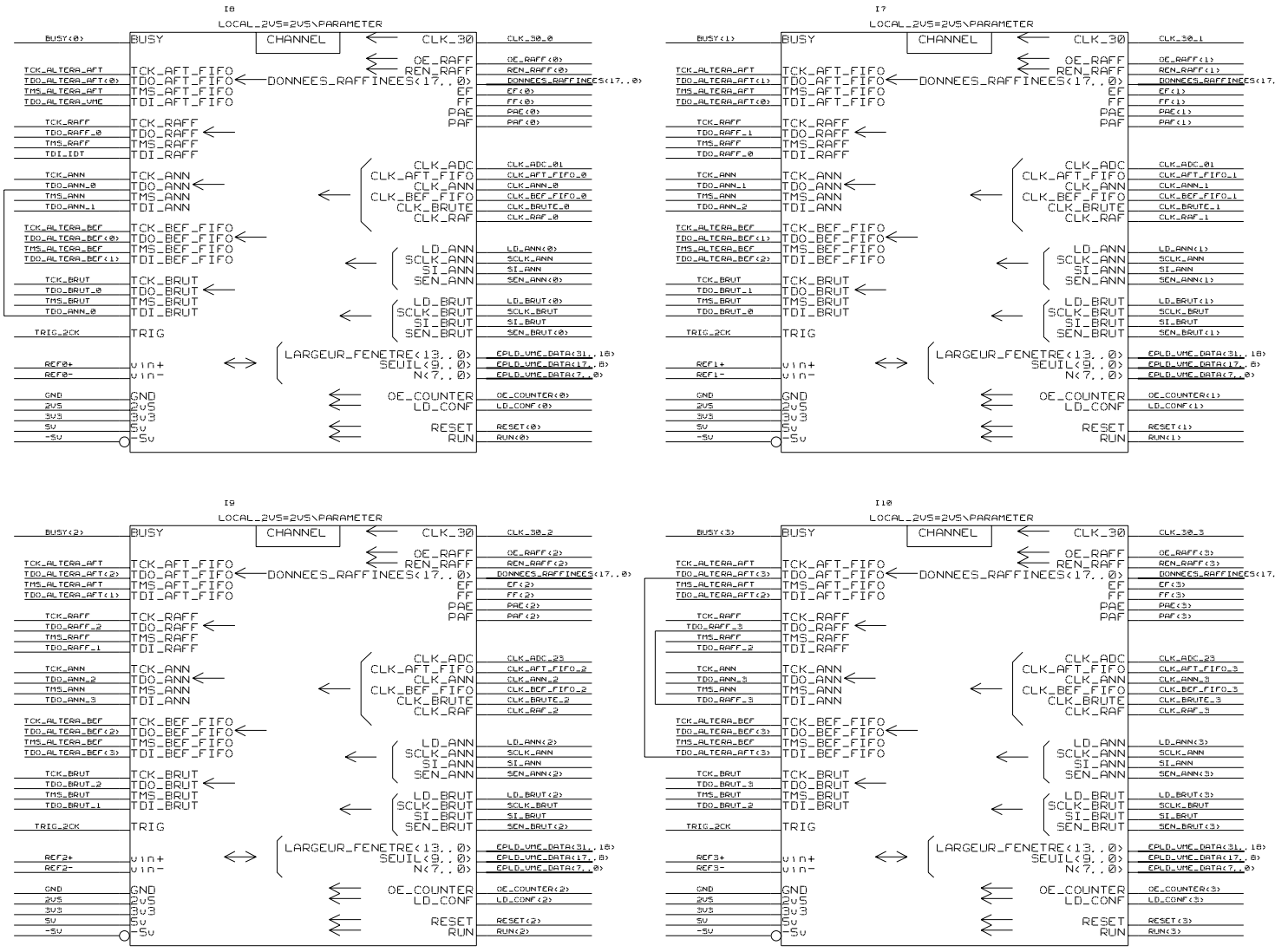


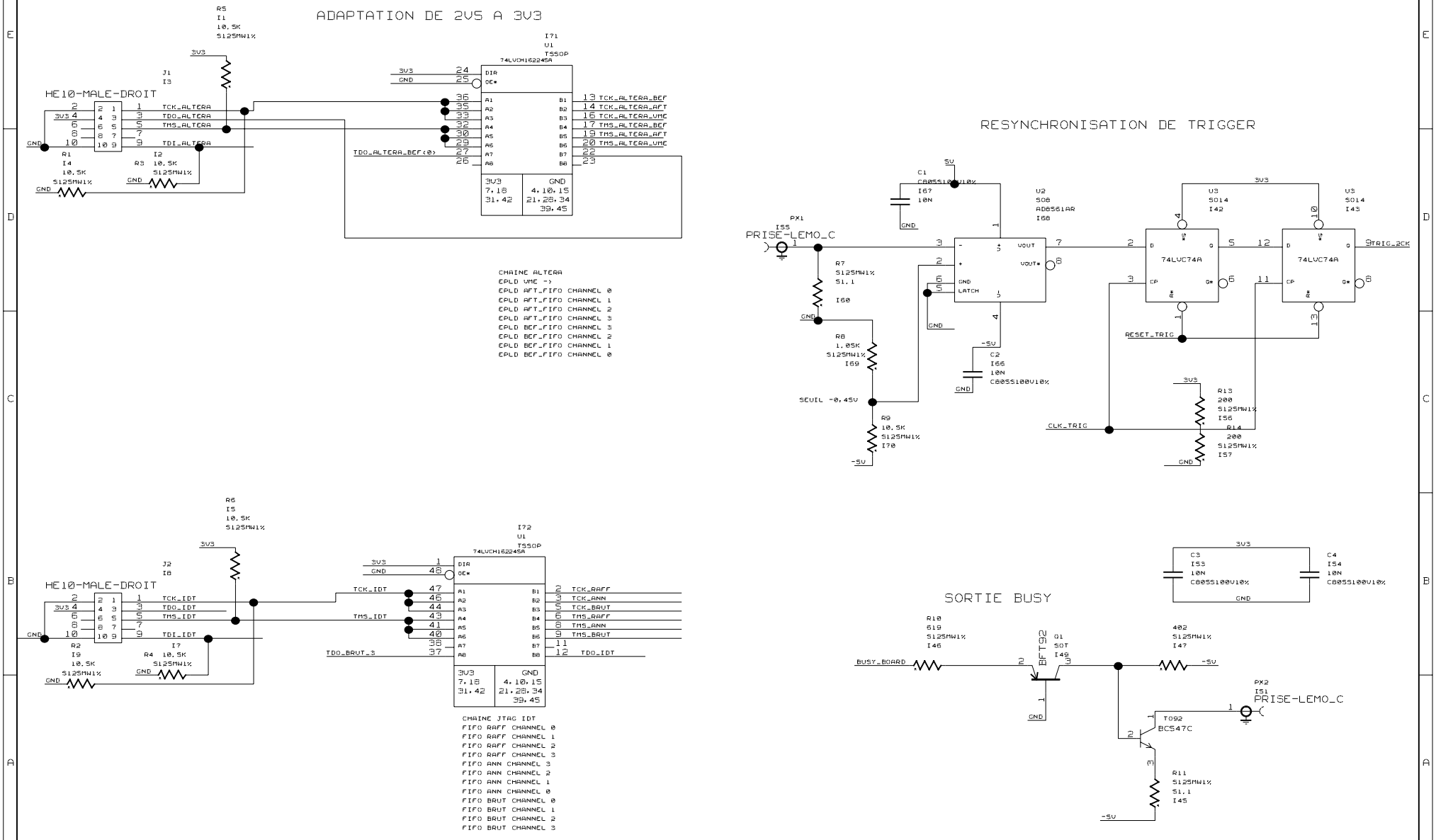
REF: VERSION 2		
	ETUDE: RH_FLASH	PAGE:
	DESSIN:O. BOURRION	DATE:

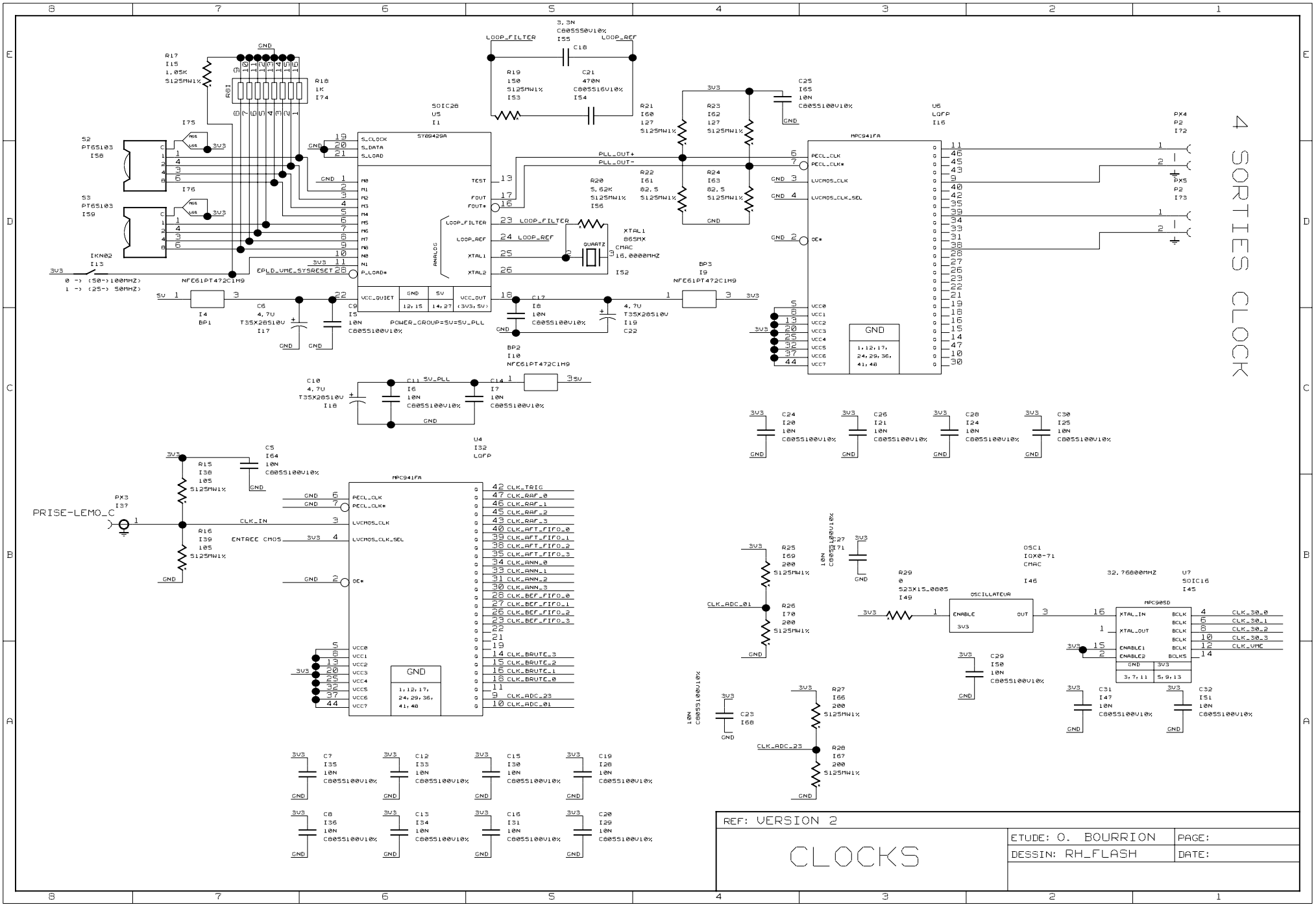

IN+
IN-
VIN+
VIN-
ANALOG_IN+
ANALOG_IN-
RESITANCE 50 OHMS AU PLUS PRES DES AOP
OFFSET -2V5, 0V
SORTIE=IN-OFFSET
SOIC
AD8062
NFE61PT472C1H9
SV100ELT22
AD9433BSQ
LQFP_ED
ADC_ANALOG+
ADC_ANALOG-
ADC_ECL+
ADC_ECL-
Z0=50 OHMS
CLK_ADC 6
MODE BINAIRE
UNE RESISTANCE NON MONTE
VREF_ADC
DEFAULT
verte
BSS138
RUN
-5V
5V
3V3
2V5
GND
TITLE: RH_FLASH
VERSION 2
DATE:
ENGINEER: O. BOURRION
PAGE:

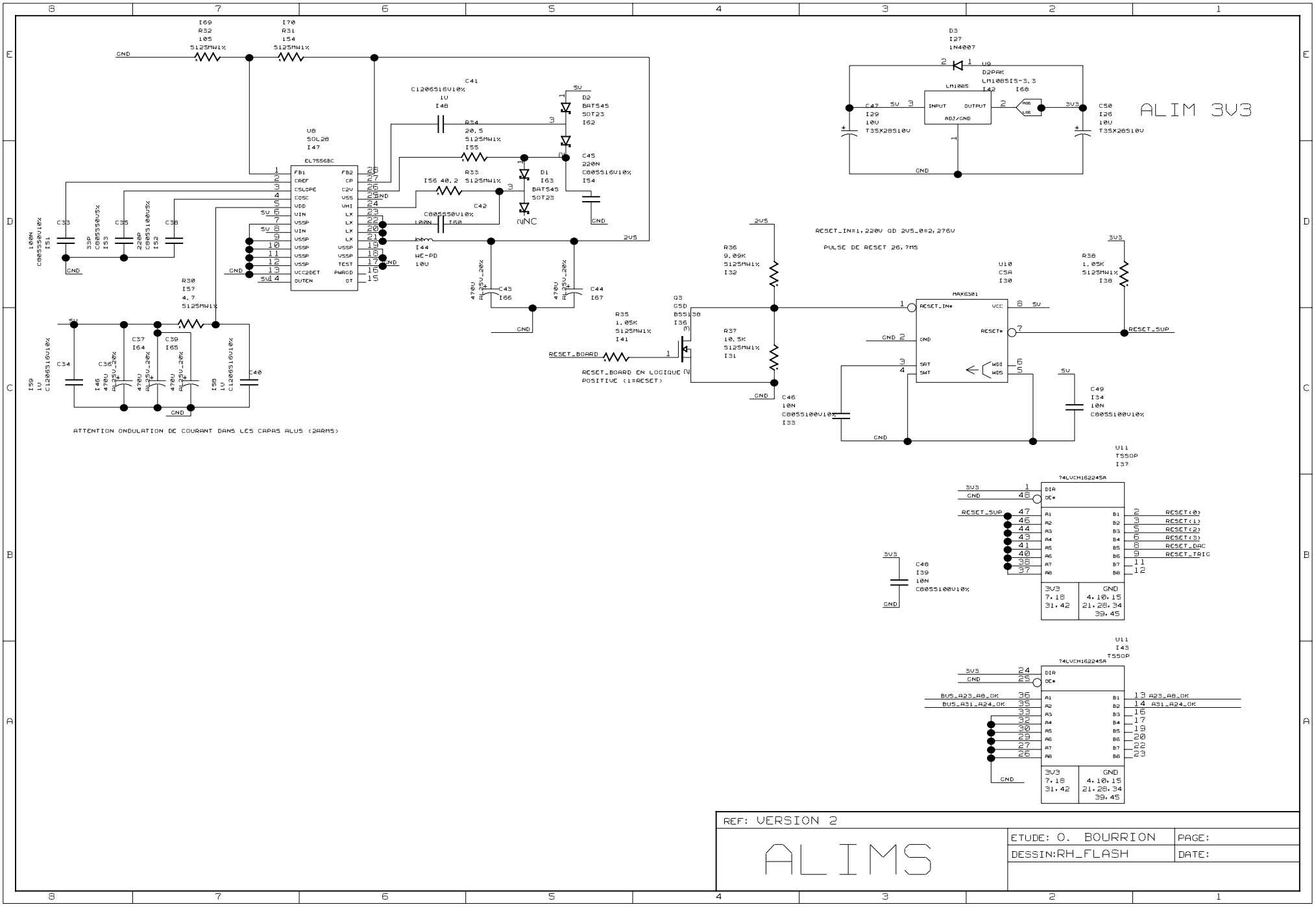

FIFO_BRUTE
U29
BGA144
IDT72T1885L6-7BB
I1
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
BEF_FIFO
U30
TQFP
I27
POWER_GROUP=2V5=%LOCAL_2V5
EPM7256BTC100
FIFO_ANNEXE
U31
BGA144
IDT72T1845L6-7BB
I81
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
TDO_BRUT
TDO_BEF_FIFO
TDO_ANN
CLK_BRUTE
CLK_ANN
CLK_BEF_FIFO
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:

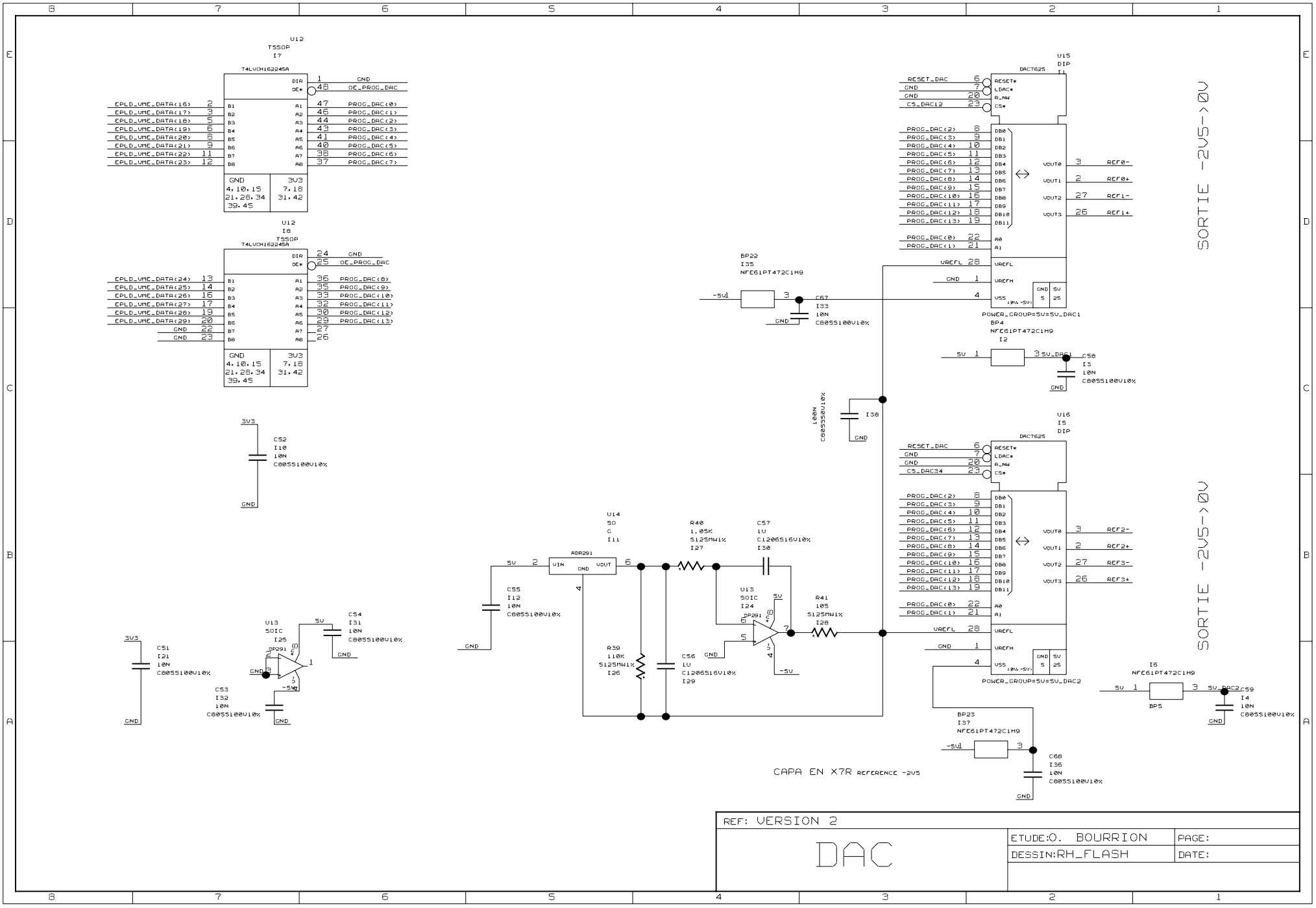

AFT_FIFO
FIFO_RAFFINEE
U32
TQFP
POWER_GROUP=2V5=%LOCAL_2V5
U33
BGA144
IDT72T1885L6-7BB
POWER_GROUP=2V5=%LOCAL_2V5
IDT72T
CLK_AFT_FIFO
RESET
TDI_AFT_FIFO
TMS_AFT_FIFO
TCK_AFT_FIFO
TDO_AFT_FIFO
CLK_RAF
TCK_RAFF
TMS_RAFF
TDI_RAFF
TDO_RAFF
CLK_30
WEN_RAFF
REN_RAFF
OE_RAFF
SERIAL MODE
IDT STANDART
SYNCHRONOUS
N=M=1023
RUN
TEST
GND
2V5
3V3
REF: VERSION 2
ETUDE: RH_FLASH
PAGE:
DESSIN:O. BOURRION
DATE:

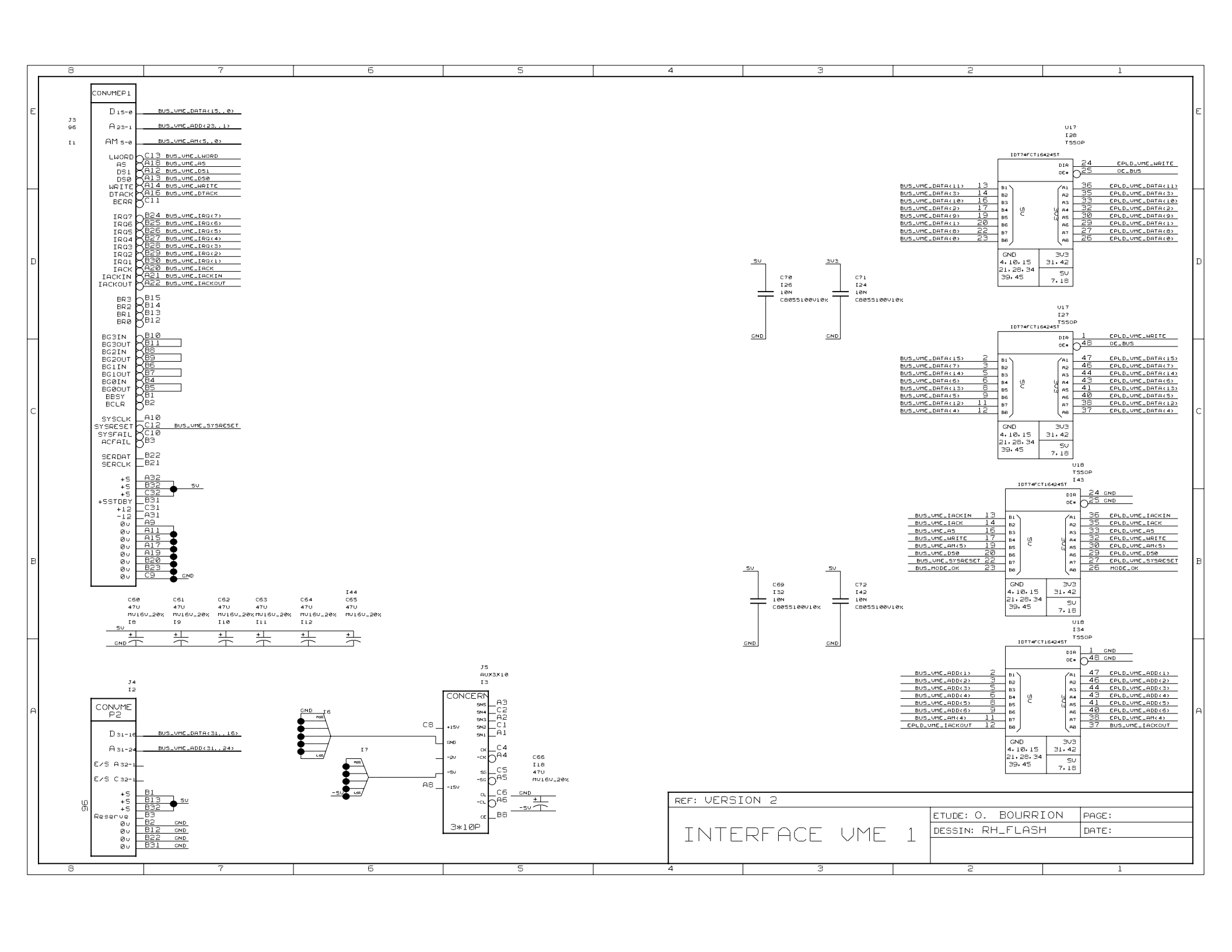

I8
LOCAL_2V5=2V5\PARAMETER
I7
LOCAL_2V5=2V5\PARAMETER
I9
LOCAL_2V5=2V5\PARAMETER
I10
LOCAL_2V5=2V5\PARAMETER
CHANNEL
LARGEUR_FENETRE<13,,0>
SEUIL<9,,0>
N<7,,0>
DONNEES_RAFFINEES<17,,0>
REF: VERSION 2
VOIES
ETUDE: O. BOURRION
DESSIN: RH_FLASH
PAGE:
DATE:

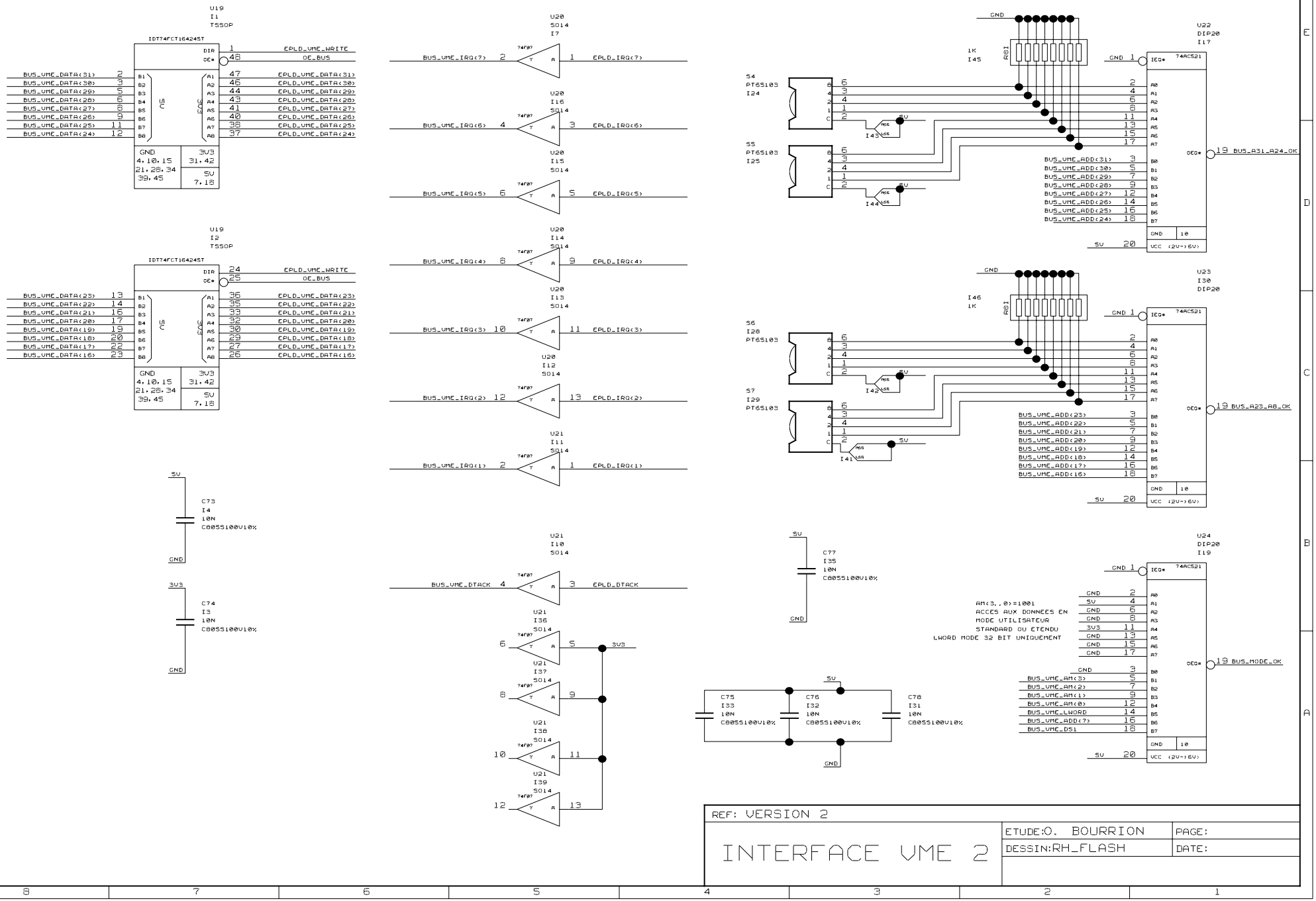

ADAPTATION DE 2V5 A 3V3
HE10-MALE-DROIT
74LVCH162245A
TCK_ALTERA
TDO_ALTERA
TMS_ALTERA
TDI_ALTERA
TCK_ALTERA_BEF
TCK_ALTERA_AFT
TCK_ALTERA_VME
TMS_ALTERA_BEF
TMS_ALTERA_AFT
TMS_ALTERA_VME
TDO_ALTERA_BEF(0)
CHAINE ALTERA
EPLD VME ->
EPLD AFT_FIFO CHANNEL 0
EPLD AFT_FIFO CHANNEL 1
EPLD AFT_FIFO CHANNEL 2
EPLD AFT_FIFO CHANNEL 3
EPLD BEF_FIFO CHANNEL 3
EPLD BEF_FIFO CHANNEL 2
EPLD BEF_FIFO CHANNEL 1
EPLD BEF_FIFO CHANNEL 0
RESYNCHRONISATION DE TRIGGER
PRISE-LEMO_C
74LVC74A
RESET_TRIG
CLK_TRIG
STRIG_2CK
SEUIL -0,45V
SORTIE BUSY
BUSY_BOARD
BFT92
BC547C
PRISE-LEMO_C
HE10-MALE-DROIT
74LVCH162245A
TCK_IDT
TDO_IDT
TMS_IDT
TDI_IDT
TCK_RAFF
TCK_ANN
TCK_BRUT
TMS_RAFF
TMS_ANN
TMS_BRUT
TDO_BRUT_3
CHAINE JTAG IDT
FIFO RAFF CHANNEL 0
FIFO RAFF CHANNEL 1
FIFO RAFF CHANNEL 2
FIFO RAFF CHANNEL 3
FIFO ANN CHANNEL 3
FIFO ANN CHANNEL 2
FIFO ANN CHANNEL 1
FIFO ANN CHANNEL 0
FIFO BRUT CHANNEL 0
FIFO BRUT CHANNEL 1
FIFO BRUT CHANNEL 2
FIFO BRUT CHANNEL 3
REF: VERSION 2
JTAG + TRIG
ETUDE:O. BOURRION
DESSIN: RH_FLASH
PAGE:
DATE:

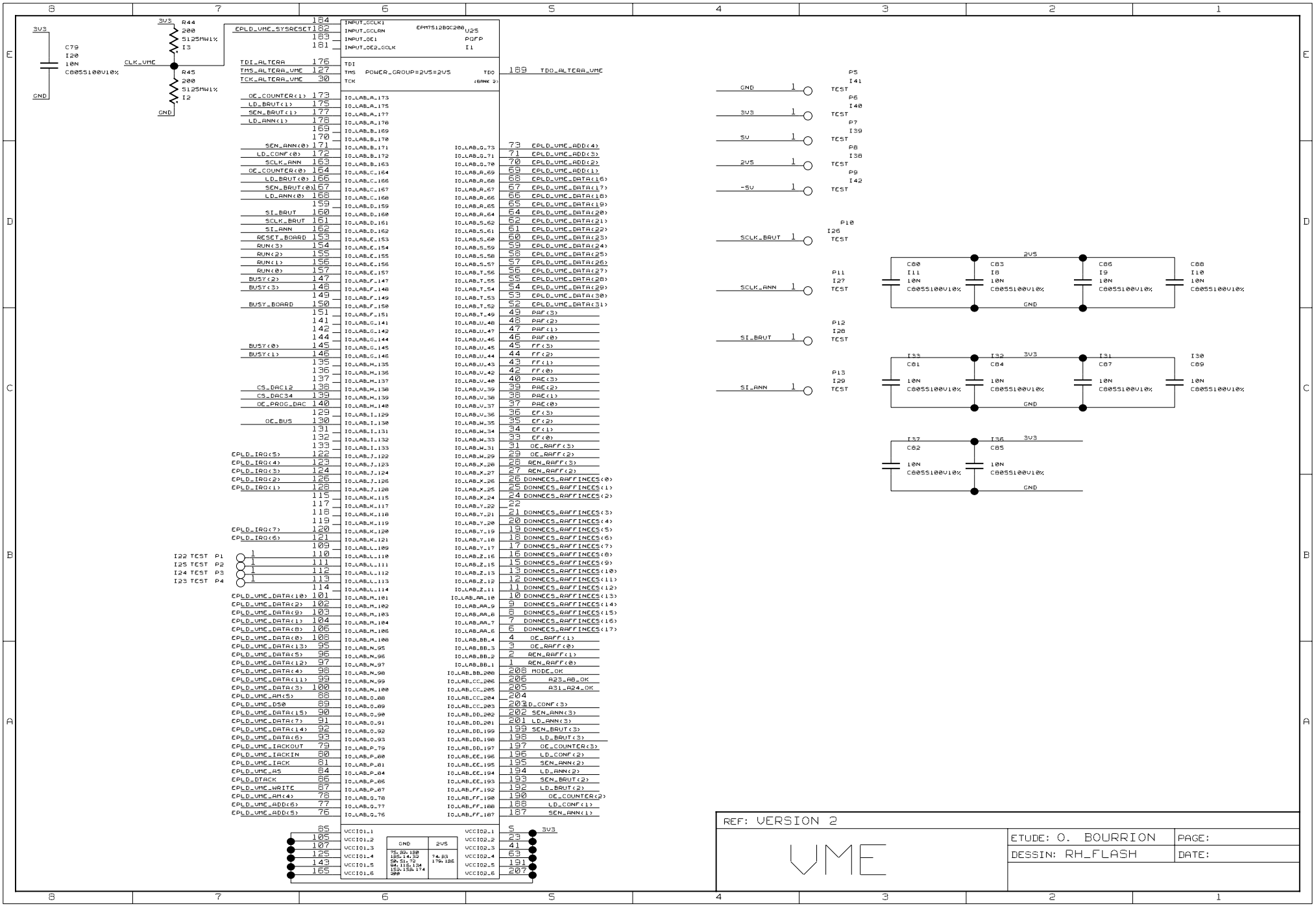

4 SORTIES CLOCK
SOIC28
U5
I1
SY89429A
S_CLOCK
S_DATA
S_LOAD
TEST
FOUT
FOUT*
LOOP_FILTER
LOOP_REF
XTAL1
XTAL2
P_LOAD*
ANALOG
VCC_QUIET
GND
5V
VCC_OUT
12, 15
14, 27
(3V3, 5V)
POWER_GROUP=5V=5V_PLL
EPLD_VME_SYSRESET
IKN02
I13
0 -> (50->100MHZ)
1 -> (25-> 50MHZ)
S2
PT65103
I58
S3
PT65103
I59
I75
I76
R17
I15
1.05K
S125MW1%
R18
1K
I74
RB1
XTAL1
865MX
CMAC
16.0000MHZ
QUARTZ
I52
R19
I50
S125MW1%
I53
C18
3,3N
C0805S0V10%
I55
C21
470N
C0805S16V10%
I54
R20
5.62K
S125MW1%
I56
R21
I60
127
S125MW1%
R22
I61
82.5
S125MW1%
R23
I62
127
S125MW1%
R24
I63
82.5
S125MW1%
C25
I65
10N
C0805S100V10%
MPC941FA
PECL_CLK
PECL_CLK*
LVCMOS_CLK
LVCMOS_CLK_SEL
OE*
VCC0
VCC1
VCC2
VCC3
VCC4
VCC5
VCC6
VCC7
1, 12, 17, 24, 29, 36, 41, 48
U6
LQFP
I16
PX4
P2
I72
PX5
P2
I73
BP1
I4
C6
4,7U
T35X20S10V
I17
C9
I5
10N
C0805S100V10%
BP2
I10
NFE61PT472C1H9
C17
I8
10N
C0805S100V10%
BP3
I9
NFE61PT472C1H9
C22
4,7U
T35X20S10V
I19
C10
4,7U
T35X20S10V
I18
C11
5V_PLL
I6
10N
C0805S100V10%
C14
I7
10N
C0805S100V10%
C24
I20
10N
C0805S100V10%
C26
I21
10N
C0805S100V10%
C28
I24
10N
C0805S100V10%
C30
I25
10N
C0805S100V10%
U4
I32
LQFP
C5
I64
10N
C0805S100V10%
R15
I38
105
S125MW1%
R16
I39
105
S125MW1%
PX3
I37
PRISE-LEMO_C
CLK_IN
ENTREE CMOS
CLK_TRIG
CLK_RAF_0
CLK_RAF_1
CLK_RAF_2
CLK_RAF_3
CLK_AFT_FIFO_0
CLK_AFT_FIFO_1
CLK_AFT_FIFO_2
CLK_AFT_FIFO_3
CLK_ANN_0
CLK_ANN_1
CLK_ANN_2
CLK_ANN_3
CLK_BEF_FIFO_0
CLK_BEF_FIFO_1
CLK_BEF_FIFO_2
CLK_BEF_FIFO_3
CLK_BRUTE_3
CLK_BRUTE_2
CLK_BRUTE_1
CLK_BRUTE_0
CLK_ADC_23
CLK_ADC_01
R25
I69
200
S125MW1%
R26
I70
200
S125MW1%
R27
I71
200
S125MW1%
R28
I67
200
S125MW1%
C23
I68
10N
C0805S100V10%
R29
0
S23X15_0805
I49
OSCILLATEUR
ENABLE
OUT
3V3
OSC1
IQX0-71
CMAC
I46
32,768000MHZ
U7
SOIC16
I45
MPC905D
XTAL_IN
XTAL_OUT
ENABLE1
ENABLE2
BCLK
BCLK5
3, 7, 11
5, 9, 13
CLK_30_0
CLK_30_1
CLK_30_2
CLK_30_3
CLK_VME
C29
I50
10N
C0805S100V10%
C31
I47
10N
C0805S100V10%
C32
I51
10N
C0805S100V10%
C7
I35
10N
C0805S100V10%
C12
I33
10N
C0805S100V10%
C15
I30
10N
C0805S100V10%
C19
I26
10N
C0805S100V10%
C8
I36
10N
C0805S100V10%
C13
I34
10N
C0805S100V10%
C16
I31
10N
C0805S100V10%
C20
I29
10N
C0805S100V10%
REF: VERSION 2
CLOCKS
ETUDE: O. BOURRION
DESSIN: RH_FLASH
PAGE:
DATE:


ALIM 3V3
RESET_IN=1,220V OU 2V5_0=2,276V
PULSE DE RESET 26,7MS
RESET_BOARD EN LOGIQUE POSITIVE (1=RESET)
ATTENTION ONDULATION DE COURANT DANS LES CAPAS ALUS (2ARMS)
REF: VERSION 2
ALIMS
ETUDE: O. BOURRION
DESSIN:RH_FLASH
PAGE:
DATE:


U12
TSSOP
I7
74LVCH162245A
DIR
OE*
EPLD_VME_DATA<16>
EPLD_VME_DATA<17>
EPLD_VME_DATA<18>
EPLD_VME_DATA<19>
EPLD_VME_DATA<20>
EPLD_VME_DATA<21>
EPLD_VME_DATA<22>
EPLD_VME_DATA<23>
B1
B2
B3
B4
B5
B6
B7
B8
A1
A2
A3
A4
A5
A6
A7
A8
GND
OE_PROG_DAC
PROG_DAC<0>
PROG_DAC<1>
PROG_DAC<2>
PROG_DAC<3>
PROG_DAC<4>
PROG_DAC<5>
PROG_DAC<6>
PROG_DAC<7>
GND
4,10,15
21,28,34
39,45
3V3
7,18
31,42
U12
I8
TSSOP
74LVCH162245A
EPLD_VME_DATA<24>
EPLD_VME_DATA<25>
EPLD_VME_DATA<26>
EPLD_VME_DATA<27>
EPLD_VME_DATA<28>
EPLD_VME_DATA<29>
PROG_DAC<8>
PROG_DAC<9>
PROG_DAC<10>
PROG_DAC<11>
PROG_DAC<12>
PROG_DAC<13>
U15
DIP
I1
DAC7625
RESET_DAC
CS_DAC12
RESET*
LDAC*
R_W
CS*
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
DB8
DB9
DB10
DB11
VOUT0
VOUT1
VOUT2
VOUT3
REF0-
REF0+
REF1-
REF1+
A0
A1
VREFL
VREFH
VSS
-5V
5V
POWER_GROUP=5V=5V_DAC1
BP4
NFC61PT472C1H9
I2
5V_DAC1
C58
I3
10N
C0805100V10%
BP22
I35
C67
I33
C58
I38
U16
I5
CS_DAC34
REF2-
REF2+
REF3-
REF3+
POWER_GROUP=5V=5V_DAC2
BP5
I6
5V_DAC2
C59
I4
BP23
I37
C68
I36
U14
SO
C
I11
ADR291
VIN
VOUT
R40
1.05K
S125MW1%
I27
CS7
1U
C1206S16V10%
I30
CS5
I12
R41
105
S125MW1%
I28
U13
SOIC
I24
OP291
R39
110K
S125MW1%
I26
CS6
1U
C1206S16V10%
I29
U13
SOIC
I25
CS4
I31
CS1
I21
CS3
I32
CS2
I10
SORTIE -2V5->0V
SORTIE -2V5->0V
CAPA EN X7R REFERENCE -2V5
REF: VERSION 2
DAC
ETUDE:O. BOURRION
PAGE:
DESSIN:RH_FLASH
DATE:


CONUMEP1
J3
96
I1
D15-0
BUS_VME_DATA<15..0>
A23-1
BUS_VME_ADD<23..1>
AM5-0
BUS_VME_AM<5..0>
LWORD C13 BUS_VME_LWORD
AS A16 BUS_VME_AS
DS1 A12 BUS_VME_DS1
DS0 A13 BUS_VME_DS0
WRITE A14 BUS_VME_WRITE
DTACK A16 BUS_VME_DTACK
BERR C11
IRQ7 B24 BUS_VME_IRQ<7>
IRQ6 B25 BUS_VME_IRQ<6>
IRQ5 B26 BUS_VME_IRQ<5>
IRQ4 B27 BUS_VME_IRQ<4>
IRQ3 B28 BUS_VME_IRQ<3>
IRQ2 B29 BUS_VME_IRQ<2>
IRQ1 B30 BUS_VME_IRQ<1>
IACK A20 BUS_VME_IACK
IACKIN A21 BUS_VME_IACKIN
IACKOUT A22 BUS_VME_IACKOUT
BR3 B15
BR2 B14
BR1 B13
BR0 B12
BG3IN B10
BG3OUT B11
BG2IN B8
BG2OUT B9
BG1IN B6
BG1OUT B7
BG0IN B4
BG0OUT B5
BBSY B1
BCLR B2
SYSCLK A10
SYSRESET C12 BUS_VME_SYSRESET
SYSFAIL C10
ACFAIL B3
SERDAT B22
SERCLK B21
+5 A32
+5 B32
+5 C32
5V
+5STDBY B31
+12 C31
-12 A31
0V A9
0V A11
0V A15
0V A17
0V A19
0V B20
0V B23
0V C9
GND
C60 47U MV16V_20% I8
C61 47U MV16V_20% I9
C62 47U MV16V_20% I10
C63 47U MV16V_20% I11
C64 47U MV16V_20% I12
C65 47U MV16V_20% I44
J4
I2
CONVME P2
D31-16
BUS_VME_DATA<31..16>
A31-24
BUS_VME_ADD<31..24>
E/S A32-1
E/S C32-1
+5 B1
+5 B13
+5 B32
Reserve B3
0V B2 GND
0V B12 GND
0V B22 GND
0V B31 GND
96
J5
AUX3X10
I3
CONCERN
3*10P
C8 +15V
A8 -15V
GND
-2V
-5V
I6
I7
C66 I18 47U MV16V_20%
U17 I28 TSSOP
U17 I27 TSSOP
U18 I43 TSSOP
U18 I34 TSSOP
IDT74FCT16424ST
DIR
OE*
5V
EOE
GND 4,10,15 21,28,34 39,45
3V3 31,42
5V 7,18
C70 I26 10N C0805S100V10%
C71 I24 10N C0805S100V10%
C69 I32 10N C0805S100V10%
C72 I42 10N C0805S100V10%
EPLD_VME_WRITE
OE_BUS
EPLD_VME_DATA<11>
EPLD_VME_DATA<3>
EPLD_VME_DATA<10>
EPLD_VME_DATA<2>
EPLD_VME_DATA<9>
EPLD_VME_DATA<1>
EPLD_VME_DATA<8>
EPLD_VME_DATA<0>
EPLD_VME_DATA<15>
EPLD_VME_DATA<7>
EPLD_VME_DATA<14>
EPLD_VME_DATA<6>
EPLD_VME_DATA<13>
EPLD_VME_DATA<5>
EPLD_VME_DATA<12>
EPLD_VME_DATA<4>
EPLD_VME_IACKIN
EPLD_VME_IACK
EPLD_VME_AS
EPLD_VME_WRITE
EPLD_VME_AM<5>
EPLD_VME_DS0
EPLD_VME_SYSRESET
MODE_OK
EPLD_VME_ADD<1>
EPLD_VME_ADD<2>
EPLD_VME_ADD<3>
EPLD_VME_ADD<4>
EPLD_VME_ADD<5>
EPLD_VME_ADD<6>
EPLD_VME_AM<4>
BUS_VME_IACKOUT
REF: VERSION 2
INTERFACE VME 1
ETUDE: O. BOURRION
DESSIN: RH_FLASH
PAGE:
DATE:


U19
I1
TSSOP
IDT74FCT164245T
DIR
OE*
BUS_VME_DATA<31>
BUS_VME_DATA<30>
BUS_VME_DATA<29>
BUS_VME_DATA<28>
BUS_VME_DATA<27>
BUS_VME_DATA<26>
BUS_VME_DATA<25>
BUS_VME_DATA<24>
EPLD_VME_WRITE
OE_BUS
EPLD_VME_DATA<31>
EPLD_VME_DATA<30>
EPLD_VME_DATA<29>
EPLD_VME_DATA<28>
EPLD_VME_DATA<27>
EPLD_VME_DATA<26>
EPLD_VME_DATA<25>
EPLD_VME_DATA<24>
GND
4,10,15
21,28,34
39,45
3V3
31,42
5V
7,18
U19
I2
TSSOP
IDT74FCT164245T
BUS_VME_DATA<23>
BUS_VME_DATA<22>
BUS_VME_DATA<21>
BUS_VME_DATA<20>
BUS_VME_DATA<19>
BUS_VME_DATA<18>
BUS_VME_DATA<17>
BUS_VME_DATA<16>
EPLD_VME_WRITE
OE_BUS
EPLD_VME_DATA<23>
EPLD_VME_DATA<22>
EPLD_VME_DATA<21>
EPLD_VME_DATA<20>
EPLD_VME_DATA<19>
EPLD_VME_DATA<18>
EPLD_VME_DATA<17>
EPLD_VME_DATA<16>
GND
4,10,15
21,28,34
39,45
3V3
31,42
5V
7,18
5V
C73
I4
10N
C0805100V10%
GND
3V3
C74
I3
10N
C0805100V10%
GND
U20
SO14
I7
BUS_VME_IRQ<7>
EPLD_IRQ<7>
U20
I16
SO14
BUS_VME_IRQ<6>
EPLD_IRQ<6>
U20
I15
SO14
BUS_VME_IRQ<5>
EPLD_IRQ<5>
U20
I14
SO14
BUS_VME_IRQ<4>
EPLD_IRQ<4>
U20
I13
SO14
BUS_VME_IRQ<3>
EPLD_IRQ<3>
U20
I12
SO14
BUS_VME_IRQ<2>
EPLD_IRQ<2>
U21
I11
SO14
BUS_VME_IRQ<1>
EPLD_IRQ<1>
U21
I10
SO14
BUS_VME_DTACK
EPLD_DTACK
U21
I36
SO14
U21
I37
SO14
U21
I38
SO14
U21
I39
SO14
3V3
GND
U22
DIP20
I17
1K
I45
RB1
S4
PT65103
I24
S5
PT65103
I25
I43
I44
5V
BUS_VME_ADD<31>
BUS_VME_ADD<30>
BUS_VME_ADD<29>
BUS_VME_ADD<28>
BUS_VME_ADD<27>
BUS_VME_ADD<26>
BUS_VME_ADD<25>
BUS_VME_ADD<24>
74ACS21
OEG*
19 BUS_A31_A24_OK
GND
10
VCC (2V->6V)
U23
I30
DIP20
I46
1K
RB1
S6
I28
PT65103
S7
I29
PT65103
I41
I42
BUS_VME_ADD<23>
BUS_VME_ADD<22>
BUS_VME_ADD<21>
BUS_VME_ADD<20>
BUS_VME_ADD<19>
BUS_VME_ADD<18>
BUS_VME_ADD<17>
BUS_VME_ADD<16>
19 BUS_A23_A8_OK
U24
DIP20
I19
AM<3,,0>=1001
ACCES AUX DONNEES EN
MODE UTILISATEUR
STANDARD OU ETENDU
LWORD MODE 32 BIT UNIQUEMENT
BUS_VME_AM<3>
BUS_VME_AM<2>
BUS_VME_AM<1>
BUS_VME_AM<0>
BUS_VME_LWORD
BUS_VME_ADD<7>
BUS_VME_DS1
19 BUS_MODE_OK
C77
I35
10N
C0805100V10%
C75
I33
10N
C0805100V10%
C75
I32
10N
C0805100V10%
C78
I31
10N
C0805100V10%
REF: VERSION 2
INTERFACE VME 2
ETUDE:O. BOURRION
PAGE:
DESSIN:RH_FLASH
DATE:


3V3
C79
I20
10N
C0805S100V10%
GND
CLK_VME
R44
200
S125MW1%
I3
R45
200
S125MW1%
I2
GND
EPLD_VME_SYSRESET
TDI_ALTERA
TMS_ALTERA_VME
TCK_ALTERA_VME
OE_COUNTER<1>
LD_BRUT<1>
SEN_BRUT<1>
LD_ANN<1>
SEN_ANN<0>
LD_CONF<0>
SCLK_ANN
OE_COUNTER<0>
LD_BRUT<0>
SEN_BRUT<0>
LD_ANN<0>
SI_BRUT
SCLK_BRUT
SI_ANN
RESET_BOARD
RUN<3>
RUN<2>
RUN<1>
RUN<0>
BUSY<2>
BUSY<3>
BUSY_BOARD
BUSY<0>
BUSY<1>
CS_DAC12
CS_DAC34
OE_PROG_DAC
OE_BUS
EPLD_IRQ<5>
EPLD_IRQ<4>
EPLD_IRQ<3>
EPLD_IRQ<2>
EPLD_IRQ<1>
EPLD_IRQ<7>
EPLD_IRQ<6>
I22 TEST P1
I25 TEST P2
I24 TEST P3
I23 TEST P4
EPLD_VME_DATA<10>
EPLD_VME_DATA<2>
EPLD_VME_DATA<9>
EPLD_VME_DATA<1>
EPLD_VME_DATA<8>
EPLD_VME_DATA<0>
EPLD_VME_DATA<13>
EPLD_VME_DATA<5>
EPLD_VME_DATA<12>
EPLD_VME_DATA<4>
EPLD_VME_DATA<11>
EPLD_VME_DATA<3>
EPLD_VME_AM<5>
EPLD_VME_DS0
EPLD_VME_DATA<15>
EPLD_VME_DATA<7>
EPLD_VME_DATA<14>
EPLD_VME_DATA<6>
EPLD_VME_IACKOUT
EPLD_VME_IACKIN
EPLD_VME_IACK
EPLD_VME_AS
EPLD_DTACK
EPLD_VME_WRITE
EPLD_VME_AM<4>
EPLD_VME_ADD<6>
EPLD_VME_ADD<5>
INPUT_GCLK1
INPUT_GCLRN
INPUT_OE1
INPUT_OE2_GCLK
EPM7512BQC208
U25
POFP
I1
TDI
TMS
TCK
POWER_GROUP=2V5=2V5
TDO
(BANK 2)
TDO_ALTERA_VME
IO_LAB_A_173
IO_LAB_A_175
IO_LAB_A_177
IO_LAB_A_178
IO_LAB_B_169
IO_LAB_B_170
IO_LAB_B_171
IO_LAB_B_172
IO_LAB_B_163
IO_LAB_C_164
IO_LAB_C_165
IO_LAB_C_167
IO_LAB_C_168
IO_LAB_D_159
IO_LAB_D_160
IO_LAB_D_161
IO_LAB_D_162
IO_LAB_E_153
IO_LAB_E_154
IO_LAB_E_155
IO_LAB_E_156
IO_LAB_E_157
IO_LAB_F_147
IO_LAB_F_148
IO_LAB_F_149
IO_LAB_F_150
IO_LAB_F_151
IO_LAB_G_141
IO_LAB_G_142
IO_LAB_G_144
IO_LAB_G_145
IO_LAB_G_146
IO_LAB_H_135
IO_LAB_H_136
IO_LAB_H_137
IO_LAB_H_138
IO_LAB_H_139
IO_LAB_H_140
IO_LAB_I_129
IO_LAB_I_130
IO_LAB_I_131
IO_LAB_I_132
IO_LAB_I_133
IO_LAB_J_122
IO_LAB_J_123
IO_LAB_J_124
IO_LAB_J_125
IO_LAB_J_126
IO_LAB_K_115
IO_LAB_K_117
IO_LAB_K_118
IO_LAB_K_119
IO_LAB_K_120
IO_LAB_K_121
IO_LAB_L_109
IO_LAB_L_110
IO_LAB_L_111
IO_LAB_L_112
IO_LAB_L_113
IO_LAB_L_114
IO_LAB_M_101
IO_LAB_M_102
IO_LAB_M_103
IO_LAB_M_104
IO_LAB_M_105
IO_LAB_M_106
IO_LAB_N_95
IO_LAB_N_96
IO_LAB_N_97
IO_LAB_N_98
IO_LAB_N_99
IO_LAB_N_100
IO_LAB_O_88
IO_LAB_O_89
IO_LAB_O_90
IO_LAB_O_91
IO_LAB_O_92
IO_LAB_O_93
IO_LAB_P_79
IO_LAB_P_80
IO_LAB_P_81
IO_LAB_P_84
IO_LAB_P_86
IO_LAB_P_87
IO_LAB_Q_76
IO_LAB_Q_77
IO_LAB_Q_78
IO_LAB_G_73
IO_LAB_G_71
IO_LAB_G_70
IO_LAB_A_69
IO_LAB_A_68
IO_LAB_A_67
IO_LAB_A_66
IO_LAB_A_65
IO_LAB_A_64
IO_LAB_S_62
IO_LAB_S_61
IO_LAB_S_60
IO_LAB_S_59
IO_LAB_S_58
IO_LAB_S_57
IO_LAB_T_56
IO_LAB_T_55
IO_LAB_T_54
IO_LAB_T_53
IO_LAB_T_52
IO_LAB_T_49
IO_LAB_U_48
IO_LAB_U_47
IO_LAB_U_46
IO_LAB_U_45
IO_LAB_U_44
IO_LAB_U_43
IO_LAB_V_42
IO_LAB_V_40
IO_LAB_V_39
IO_LAB_V_38
IO_LAB_V_37
IO_LAB_V_36
IO_LAB_W_35
IO_LAB_W_34
IO_LAB_W_33
IO_LAB_W_31
IO_LAB_W_29
IO_LAB_X_28
IO_LAB_X_27
IO_LAB_X_26
IO_LAB_X_25
IO_LAB_X_24
IO_LAB_Y_22
IO_LAB_Y_21
IO_LAB_Y_20
IO_LAB_Y_19
IO_LAB_Y_18
IO_LAB_Y_17
IO_LAB_Z_16
IO_LAB_Z_15
IO_LAB_Z_13
IO_LAB_Z_12
IO_LAB_Z_11
IO_LAB_AA_10
IO_LAB_AA_9
IO_LAB_AA_8
IO_LAB_AA_7
IO_LAB_AA_6
IO_LAB_BB_4
IO_LAB_BB_3
IO_LAB_BB_2
IO_LAB_BB_1
IO_LAB_BB_208
IO_LAB_CC_206
IO_LAB_CC_205
IO_LAB_CC_204
IO_LAB_CC_203
IO_LAB_DD_202
IO_LAB_DD_201
IO_LAB_DD_199
IO_LAB_DD_198
IO_LAB_DD_197
IO_LAB_EE_196
IO_LAB_EE_195
IO_LAB_EE_194
IO_LAB_EE_193
IO_LAB_FF_192
IO_LAB_FF_190
IO_LAB_FF_188
IO_LAB_FF_187
EPLD_VME_ADD<4>
EPLD_VME_ADD<3>
EPLD_VME_ADD<2>
EPLD_VME_ADD<1>
EPLD_VME_DATA<16>
EPLD_VME_DATA<17>
EPLD_VME_DATA<18>
EPLD_VME_DATA<19>
EPLD_VME_DATA<20>
EPLD_VME_DATA<21>
EPLD_VME_DATA<22>
EPLD_VME_DATA<23>
EPLD_VME_DATA<24>
EPLD_VME_DATA<25>
EPLD_VME_DATA<26>
EPLD_VME_DATA<27>
EPLD_VME_DATA<28>
EPLD_VME_DATA<29>
EPLD_VME_DATA<30>
EPLD_VME_DATA<31>
PAF<3>
PAF<2>
PAF<1>
PAF<0>
FF<3>
FF<2>
FF<1>
FF<0>
PAE<3>
PAE<2>
PAE<1>
PAE<0>
EF<3>
EF<2>
EF<1>
EF<0>
OE_RAFF<3>
OE_RAFF<2>
REN_RAFF<3>
REN_RAFF<2>
DONNEES_RAFFINEES<0>
DONNEES_RAFFINEES<1>
DONNEES_RAFFINEES<2>
DONNEES_RAFFINEES<3>
DONNEES_RAFFINEES<4>
DONNEES_RAFFINEES<5>
DONNEES_RAFFINEES<6>
DONNEES_RAFFINEES<7>
DONNEES_RAFFINEES<8>
DONNEES_RAFFINEES<9>
DONNEES_RAFFINEES<10>
DONNEES_RAFFINEES<11>
DONNEES_RAFFINEES<12>
DONNEES_RAFFINEES<13>
DONNEES_RAFFINEES<14>
DONNEES_RAFFINEES<15>
DONNEES_RAFFINEES<16>
DONNEES_RAFFINEES<17>
OE_RAFF<1>
OE_RAFF<0>
REN_RAFF<1>
REN_RAFF<0>
MODE_OK
A23-A8_OK
A31-A24_OK
LD_CONF<3>
SEN_ANN<3>
LD_ANN<3>
SEN_BRUT<3>
LD_BRUT<3>
OE_COUNTER<3>
LD_CONF<2>
SEN_ANN<2>
LD_ANN<2>
SEN_BRUT<2>
LD_BRUT<2>
OE_COUNTER<2>
LD_CONF<1>
SEN_ANN<1>
VCCIO1_1
VCCIO1_2
VCCIO1_3
VCCIO1_4
VCCIO1_5
VCCIO1_6
GND
2V5
75.89.188
125.14.32
50.51.72
94.115.134
152.155.174
200
74.83
179.125
VCCIO2_1
VCCIO2_2
VCCIO2_3
VCCIO2_4
VCCIO2_5
VCCIO2_6
3V3
GND
5V
2V5
-5V
SCLK_BRUT
SCLK_ANN
SI_BRUT
SI_ANN
TEST
P5
I41
P6
I40
P7
I39
P8
I38
P9
I42
P10
I26
P11
I27
P12
I28
P13
I29
C80
I11
C83
I8
C85
I9
C88
I10
C81
C84
C87
C82
C85
REF: VERSION 2
VME
ETUDE: O. BOURRION
DESSIN: RH_FLASH
PAGE:
DATE: